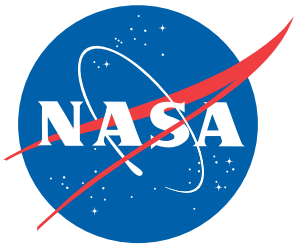
NASA/TM-2012-217576
NESC-RP-07-070



Linguistic Preprocessing and Tagging for Problem Report Trend Analysis

Robert J. Beil/NESC
Langley Research Center, Hampton, Virginia

Jane T. Malin
Johnson Space Center, Houston, Texas

May 2012

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, and organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

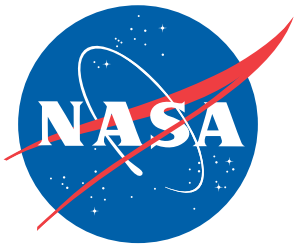
- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at 443-757-5803

- Phone the NASA STI Help Desk at 443-757-5802

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076-1320

NASA/TM-2012-217576
NESC-RP-07-070



Linguistic Preprocessing and Tagging for Problem Report Trend Analysis

Robert J. Beil/NESC
Langley Research Center, Hampton, Virginia

Jane T. Malin
Johnson Space Center, Houston, Texas

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

May 2012

Available from:

NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320
443-757-5802

Linguistic Preprocessing and Tagging for Problem Report Trend Analysis

March 29, 2012

Report Approval and Revision History

NOTE: This document was approved at the March 29, 2012, NRB. This document was submitted to the NESC Director on April 12, 2012, for configuration control.

Approved:	*Original Signature on File*	4/18/12
	NESC Director	Date

Version	Description of Revision	Office of Primary Responsibility	Effective Date
1.0	Initial Release	Mr. Robert Beil, Systems Engineering Office, Kennedy Space Center	03/29/12

Table of Contents

Technical Assessment Report

List of Figures

List of Tables

Technical Assessment Report

1.0 Notification and Authorization

Mr. Robert Beil, Systems Engineer at Kennedy Space Center (KSC), requested the NASA Engineering and Safety Center (NESC) develop a prototype tool suite that combines complementary software technology used at Johnson Space Center (JSC) and KSC for problem report preprocessing and semantic tag extraction, to improve input to data mining and trend analysis. The technology developed at JSC includes text analysis software and the Aerospace Ontology (AO), which is a nomenclature designed to support semantic analysis of aerospace problem descriptions. The KSC technology is software for natural language understanding and question answering, developed at the University of Central Florida (UCF). This combined approach will be used to analyze a variety of NASA problem reports.

An NESC out-of-board activity was approved by Ms. Dawn Schaible, Systems Engineering Office Manager, on December 7, 2007. Mr. Beil was selected to lead this assessment. Dr. Jane T. Malin (JSC), a member of the NESC Data Mining and Trending Working Group (DMTWG), is the project manager and co-principal investigator (co-PI). Dr. Fernando Gomez (UCF) is also a co-PI. Assessment plans were approved by the NESC Review Board (NRB) on December 13, 2007, and November 4, 2010. Status briefings were reviewed on February 26, 2009; December 3, 2009; and November 4, 2010.

The key stakeholders for this assessment are:

- Ms. Linda Bromley, Division Manager, Program Engineering Integration Office, JSC Engineering Directorate.
- Mr. Delmar Foster, Senior Quality Systems and Data Mining Analyst, KSC/United Space Alliance.
- Dr. Ali Shaykhian, Information Technology Relations Manager, Technical Integration Office, KSC.
- Ms. Irene Piatek, Engineering Discrepancy Report (DR) Metrics Team Lead, Systems Architecture and Integration Office, JSC Engineering Directorate.
- Dr. Jeffrey Dawson, Data Analysis and Trending, Knowledge Management Systems Office, NASA Safety Center (NSC).
- Dr. Allen Nikora, Manager, Software Element, Assurance Technology Program Office, Jet Propulsion Laboratory (JPL) Office of Safety and Mission Success (OSMS).

2.0 Signature Page

Submitted by:

Team Signature Page on File

Mr. Robert J. Beil	Date	Dr. Jane T. Malin	Date

Significant Contributors:

Mr. Land D. Fleming	Date	Dr. Fernando Gomez	Date
Dr. Carroll G. Thronesbery	Date	Dr. David R. Throop	Date

Signatories declare the findings, observations, and NESC recommendations compiled in the report are factually based from data extracted from Program/Project documents, contractor reports, and open literature, and/or generated from independently conducted tests, analysis, and inspections.

3.0 Team List

Name	Discipline	Organization
Core Team		
Bob Beil	NESC Lead	KSC
Jane Malin	Co-Principal Investigator	JSC
Fernando Gomez	Co-Principal Investigator	UCF
David Throop	Software	The Boeing Company
Land Fleming	Software-Methods	MEI Technologies®
Carroll Thronesbery	Software-ConOps	S&K Aerospace
Michel Izygon	Software-Ontology	Tietronix
Hansen Schwartz	Computer Science Graduate Student	UCF
Christopher Millward	Computer Science Graduate Student	UCF
Pam Throckmorton	MTSO Program Analyst	LaRC
Administrative Support		
Carolyn Snare	Technical Writer	LaRC/ATK

3.1 Acknowledgements

The Assessment team thanks the additional contributors who used and evaluated versions of the prototype during development: Mr. Roger G. Schwarz of the JSC Engineering Systems Architecture and Integration Office; Dr. Sheena J. Miller of the JSC Engineering Division of Software, Robotics, and Simulation; and Dr. Daniel E. Erickson and Mr. Joel M. Wilf of the JPL OSMS.

4.0 Executive Summary

4.1 Problem Description

Thousands of problem reports are generated for aerospace systems during hardware and software development, operations, and maintenance. Assessments of these problems are used to identify corrective actions to limit the potential for problem recurrence. Engineers and safety analysts need automated assistance to review large sets of problems during periodic assessments to find, verify, and assess groups of similar problem reports. Analysts have found that manual inspection with search (e.g., using cause codes or text keywords) has been impractical for large problem sets. Codes are too often misinterpreted during coding or search, or are out-of-date and inaccurate. Search and text mining can be hampered by the complexity of natural language. The results of statistical text mining often include meaningless groupings based on misleading regularities in the text.

4.2 Background

Analysts have complained about the overwhelming difficulty of reviewing large volumes of problem reports to find unknown, but important, "needles in the haystack." Text mining approaches for searching and clustering problem reports typically produced too many false alarms and too few hits. Precision and Recall are two commonly used measures of retrieval or classification accuracy. They are stated as percentages, where higher values indicate higher accuracy. Recall (i.e., percentage of all possible hits that are retrieved) has ranged from 21 to 62 percent in studies of text mining accuracy from medical and biological abstracts and facts. Precision (i.e., percentage of retrieved that are correct) is another measure of accuracy. In the application domain of problem reports, the Precision measurement is less important than Recall, because false alarm cases can be removed. Previous development of the Semantic Text Analysis Tool (STAT) and the Aerospace Ontology (AO) nomenclature had shown that analyzing and tagging the text in problem descriptions resulted in improvements in analysis of Johnson Space Center (JSC) engineering discrepancy reports (DRs). The text interpretation was used to identify metadata tags that were added to the problem report data records. These enhanced data files were reused to structure analysis of the reports. New and significant problem types were discovered by analysts using the tool.

4.3 Approach

The first solution proposed by this assessment was to improve STAT/AO retrieval accuracy and apply it to additional types of problem report data. Accuracy can be improved by using advanced parsing software for syntactic analysis to better handle the complexity of natural language problem descriptions. Integration of syntactic analysis in the Minimal Clausal

Reconstruction (MCR) algorithm from the University of Central Florida (UCF) was expected to make tagging with concepts from the AO more precise. Accurate metadata tags from STAT/AO were expected to improve the text mining results.

The second solution used in this assessment was an advanced multi-dimensional search tool. An open source tool, Flexible Information Access using Metadata in Novel Combinations (Flamenco), was enhanced to produce tables, graphs, and spreadsheets. In the enhanced Flamenco+ version, complex searches were expected to be simplified by combining codes with the hierarchically structured tags from the AO, which indicated problem types and equipment types mentioned in the problem description.

Multiple NASA Centers were involved in the process of developing and enhancing the prototypes, so that the resulting tools would be applicable across the Agency. Development of extensive user guides was planned so that the tools could be customized to increase use.

4.4 Results

In summary, the prototype tool suite improved information retrieval and text mining. Evaluations of STAT/AO tagging before and after MCR integration showed tagging accuracy for problem reports was substantially improved. Recall was improved from 10 to 86 percent, and Precision was improved from 27 to 78 percent. These accuracy levels were significant improvements over search and text mining accuracy. Adding STAT/AO tags to problem report data records for text mining further improved text mining accuracy.

Analyst effort required for trend discovery and analysis and for generation of graphs and spreadsheets of problem report trends was reduced. JSC analysts who used the prototype tool suite during this assessment found that the support for quick retrieval and inspection of groups of similar problems was beneficial. The prototype tool was used to find intractable but important topics, which have been difficult to discover and to retrieve with search methods.

The Assessment team concluded that converting problem report text into structured data can substantially reduce analysis effort while improving insight into problem-report trends. When problem-report text is converted to data by linguistic analysis and tagging, and the tagged data is used in text mining, retrieval accuracy can be significantly improved. When the tagged data is used in a modern multi-dimensional browser, analysts find it easier to search and filter problem reports and generate graphs and spreadsheets for further review. Not only is effort reduced, but there is improved insight into the problem-report data.

Based on their findings and observations, the Assessment team recommends that the NESC Data Mining and Trending Working Group (DMTWG) identify and advocate other opportunities for tool delivery and problem reporting dataset demonstrations**.** The prototype needs to be tailored for new datasets. Limited tailoring has been performed for DRs and other data sets including

problem reports, safety analyses, and requirements. To help new users, documentation and tools have been developed for tailoring AO, STAT, and Flamenco+. These have been made available as a VirtualBox® Image or as a delivery of files for downloading and installation. Extensive examples, user guides, and other documentation have been included in the delivery. These files have been transferred to Kennedy Space Center (KSC) and NASA Safety Center (NSC) users.

5.0 Assessment Plan

The goal of the proposed effort was to maximize NASA programs' safety by improving analysis of problem recurrences and similarities, using linguistic preprocessing to extract key data from problem reports. Use of semantic annotations or tags was expected to improve trend analysis effectiveness by providing more understandable output in an analyst-friendly tool suite. The goal was to develop a prototype tool suite for NASA-wide use for text mining and trending.

The objectives of the assessment included the development of a prototype tool suite that combined semantic text analysis and extraction technology being used at JSC (i.e., STAT and AO) and at KSC (i.e., natural language processing and adapted WordNet taxonomy from UCF) for problem report preprocessing and semantic tag extraction. Key extracted terms were used to improve input to data mining and trend analysis tools that process structured and unstructured data. This combined approach was used to analyze a variety of problem reports, including Space Shuttle Program problem reporting and corrective action data and JSC engineering DRs.

In the first year of this 3-year project, the Assessment team focused on a tool suite for JSC and KSC stakeholders. During the next 2 years, the scope expanded to other NASA groups addressing trends and recurrences in problem reports. Initially, the tool suite included an analyst-friendly commercial text mining and search tool. During the second and third years, the enriched output was tested in other commercial text mining tools such as the Statistical Analysis System® Text Analytics software. During the follow-on work in the fourth year, the Assessment team collaborated with users at other NASA Centers who had new types of problem-report data, missions, and analysis goals. The extraction and trend analysis suite was applied to mishap reports from the NASA Incident Reporting Information System and to the JPL Incident Surprise Anomaly problem reports. For each case, the prototype tool suite was updated to assist with problem report analyses and assessment tasks. The goal of the follow-on work was to make at least two deliveries to user groups, with associated training and support. A major addition to the suite was the use of presentation software for web-based, faceted search, and browsing to explore enriched problem-reporting data, and to interactively and automatically produce tables and trend graphs.

6.0 Problem Description and Proposed Solution

6.1 Problem Description

Engineers and safety analysts needed automated assistance to review large sets of problem reports during periodic assessments. Assessments typically target corrective actions that will limit the potential for problem recurrence. To accomplish this goal, analysts need to explore the data to identify groups of similar problem reports and then analyze them. Analysts have found that manual inspection with search (e.g., using cause codes or text keywords) has been

impractical for large problem sets. It has been known that such codes and keywords can be misinterpreted during coding and search, or are out-of-date and inaccurate. Search and text mining are hampered by the complexity of natural language descriptions of problems. The results of statistical text mining often include meaningless groupings based on misleading regularities in the text. Thus, text mining has problems with too many false alarms and too few hits. References 1 and 6 describe typical text mining results, where Recall (percentage of all possible hits that are retrieved) ranges from 21 to 62 percent.

6.2 Proposed Solution

The proposed solution is a prototype tool suite that uses advanced parsing and ontology to interpret the meaning of problem descriptions. A chart showing the tool suite is presented in Figure 6.2-1. After extracting and preprocessing the data, STAT calls a standard statistical parser and the MCR algorithm for syntactic analysis.[1] STAT uses the terms and concepts in the AO for semantic interpretation and tagging of words and phrases in the clauses.

The AO contains a hierarchical nomenclature for problem and equipment types for this purpose. The AO is a lexicalized ontology where each concept is extended with a list of words or phrases that are possible text representations of the concept. A description of AO and its development is provided in reference 3. Reference 5 describes the tool suite and its performance in detail, at a stage when the MCR had not yet been integrated. Tools and methods for tailoring the AO have been developed, based on the Protégé open source ontology development tool [ref. 7]. With these tools, analysts can systematically tailor the AO with new words and phrases that describe their specialized terminology. Analysts can define new concepts and sub-concepts in the AO, for new types of objects and problems in their domains.

STAT is designed to interpret complex natural language text and reduce false alarms and misses. Breaking out of the predefined code limitations lets problem descriptions speak for themselves. Limited codes or keywords can be replaced with metadata tags that convert text to data and identify the potentially meaningful topics in the text. Unlike codes, these tags are not required to be mutually exclusive. Multiple tags can be associated with each problem report data record to enhance search and text mining.

The prototype tool suite includes an enhanced open-source faceted browser (Flamenco+) that takes advantage of the tags. Flamenco+ was designed for flexible browsing and searching in large information spaces [ref. 2]. Browsing and filtering can be organized according to the hierarchical structures of tags (e.g., problem and equipment types). Searching can be simplified by using codes and tags. Flamenco+ was enhanced during this assessment to automatically and

[1] The MCR was developed and refined during this technical assessment [ref. 4]. The work developed by UCF on semantic interpretation in the MCR algorithm to handle verb ambiguities is provided in Appendix A.

interactively produce tables, spreadsheets, and trend graphs as the problem groups were filtered and refined.

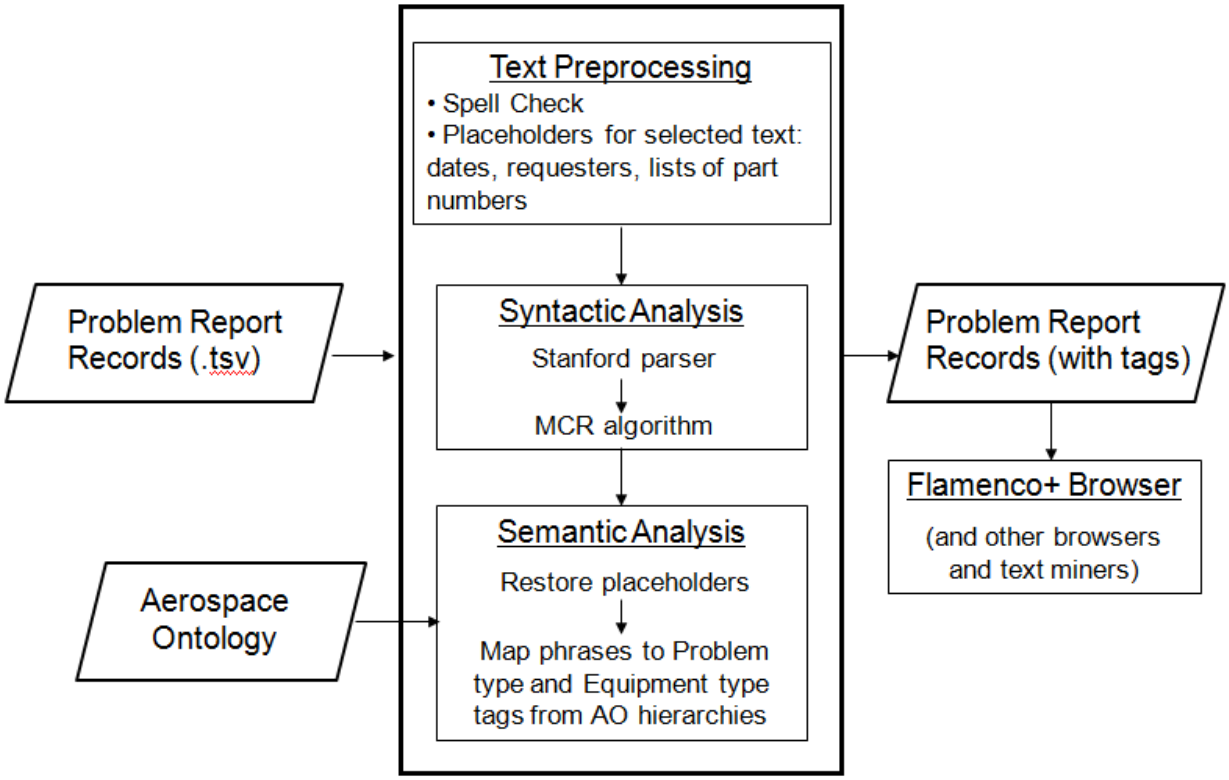


Figure 6.2-1. STAT Linguistic Analysis: Convert Text Fields into Metadata Tags

The prototype tool needs to be tailored for new datasets. Limited tailoring has been performed for DRs and to process text from other data sets including problem reports, safety analyses, and requirements. To help new users, documentation and tools have been developed for tailoring AO, STAT, and Flamenco+. These tools have been made available as a VirtualBox® Image or as a delivery of files for downloading and installation. Extensive examples, user guides, and other documentation have been included in the delivery. These files have been transferred to KSC and NSC users.

7.0 Prototype Tool Suite Evaluations

7.1 Tagging Accuracy

Evaluations of STAT before and after MCR integration showed that tagging accuracy for problem reports was substantially improved [ref. 4]. The evaluation used 36 problem types and a sample of 200 DRs from a 2007–2008 dataset. In the sample, manual scoring indicated that 101 of these reports matched at least one of the 36 problem types. Automated tagging showed that STAT provided Recall of 10 percent, and using MCR in STAT improved Recall to 86 percent. MCR in STAT improved Precision to 78 percent over Precision with STAT alone (i.e., 27 percent).

The analyst's goal was to increase Recall, to improve search by finding more true positives. Analysts find it easy to identify and eliminate a few false positives from an analysis set. The high levels of Recall for STAT with MCR compared favorably with those for the Textpresso ontology-based text miner for biological literature, which achieved 62 percent Recall about worm genomes [ref. 6]. Tagging, using MCR in STAT, compared favorably with Recall of 54 percent for search alone in a later evaluation using similar DR data (Table 7.2-1). The higher level of Recall with STAT/MCR tags shows that linguistic analysis improves search performance by finding more of the true positives that the analysts need.

7.2 Improving Text Mining Accuracy

The QuantumText text miner was selected to evaluate how including STAT tags affected text miner performance. For this evaluation, STAT was included in the MCR algorithm, as shown in Figure 6.2-1. Two thousand DR records from fiscal year 2008 were used to investigate the effect of tagging on text mining. Ten test cases (i.e., Loosely Connected, Traceability Error, Unfit, Out of Limits, Bad Identifier, Debris, Electrically Disordered, Stained, Not Aligned, and Failed Start) were drawn from the set of 36 topics selected for the previous evaluation. Each case consisted of dataset true records that were identified by STAT analysis, QuantumText search, or text mining. There were 9 to 41 true records per case, for a total of 249 true records.

The Assessment team compared three ways of retrieving the DRs: string search alone, text mining alone, and text mining with STAT tags included in the data record and double weighted. QuantumText used five exemplars (i.e., true positives) selected from the search and five non-exemplars to generate a list of DRs ranked by similarity. In scoring the text mining performance, the 50 exemplars were excluded (i.e., the five exemplars for each of the 10 cases), so that there were 199 true records remaining.

With search items removed, the task for the text miner is more difficult and accuracy scores may drop with "sure things" removed. The number of records scored was 1.5 times the number of true records found by search. In this way, the scoring took into account the varying numbers of

true records (i.e., from 9 to 41). Each case was scored and the average accuracy scores across all cases were computed. Means and ranges for Recall and Precision are shown in Table 7.2-1.

Text mining without tags produced substantially fewer true positives and substantially more false positives than search; this reduced both measures of accuracy. Alternatively, using double-weighted STAT tags compensated for these text mining problems and increased the number of total hits for analysts (i.e., 170 = 120 + 50 exemplars).

Table 7.2-1. Recall and Precision Means and Ranges

	Avg. Recall	*Avg. Precision*	$\sum T_p$	$\sum F_n$	$\sum F_p$
Search alone	0.54 (0.25-1.00)	0.88 (0.50-1.00)	126	123	25
Text Mining, No tags	0.38 (0.12-0.77)	0.37 (0.12-0.81)	81	118	110
Text Mining, STAT tags	0.66 (0.30-1.00)	0.63 (0.29-0.89)	120	79	71

Analysts can use these methods to improve trend analysis, troubleshooting, and retrospective analysis. Although the text mining results were disappointing, STAT tags improved text mining performance. More details of this evaluation are available in reference 4.

7.3 Improving Analyst Productivity

The capability to filter and cluster problem reports increases because the STAT-generated tags increase the amount of metadata and also organize it hierarchically. More and better tags can improve exploration and trend analysis. The large number of possible tags increases the possible topics that can be considered. The tags do not need to be mutually exclusive. There are more tags than the limited number of outdated and confusing codes. Typically, these codes require so much interpretation (e.g., in pull-down menus without contextual help) during coding and access that the code definitions are lost to the users. Access to the explicit meanings in the STAT tags via AO concepts helps analysts understand what makes selected groups of problem reports similar. JSC DR analysts who used the prototype tool suite during this assessment found that the support for quick retrieval and inspection of groups of similar problems was beneficial. The prototype was used to find intractable but important topics, which had been difficult to discover and retrieve with search.

Tags can be combined with codes and with search in modern faceted multi-dimensional browsers. An enhanced open-source faceted browser (i.e., Flamenco+) that takes advantage of the tags was used as part of the prototype tool suite. Using Flamenco+, browsing and filtering can be organized according to the AO hierarchical tag structure (e.g., problem and equipment

types). Searching can be simplified by using codes and tags. Flamenco+ was enhanced during this assessment to produce tables, spreadsheets, and trend graphs as a by-product of searching and browsing. Using Flamenco+, the analyst's effort required for trend analysis and generation of spreadsheets of problem reports was reduced.

8.0 Findings, Observations, and NESC Recommendations

8.1 Findings

The following findings were identified:

F-1. For problem reports, linguistic analysis and tagging, using STAT with MCR and AO improved Precision and Recall retrieval accuracy compared to search and text mining without STAT-generated tags.

F-2. Use of a modern, multi-dimensional faceted browser containing a combination of codes, keywords, and STAT-generated metadata to indicate problem types improves the capability to search, filter problem reports, and to generate graphs and spreadsheets for review.

F-3. The prototype AO, STAT, and Flamenco+ tool suite can be tailored to analyze the engineering DR database and data from other engineering projects.

8.2 Observations

The following observations were identified:

O-1. Converting problem report text into structured data substantially reduces analysis effort while improving insight into problem-report trends.

O-2. Documentation for data processing and end users was developed, so that AO, STAT, and Flamenco+ can be tailored for new datasets.

8.3 NESC Recommendations

The following NESC recommendations were identified and are directed toward the NESC DMTWG unless otherwise indicated:

R-1. Other opportunities for tool delivery, problem-reporting dataset demonstrations, and technology maturation should be identified and advocated. ***(F-1, F-2, O-1)***

R-2. Users should tailor the AO, STAT, and Flamenco+ tool suite for specific datasets. ***(F-3 and O-2)***

- Tailoring support can be obtained from the Assessment team developers.

9.0 Alternate Viewpoints

There were no alternate viewpoints identified during the course of this investigation by the Assessment team or the NRB quorum.

10.0 Other Deliverables

The STAT, AO, and Flamenco+ tools are available to NASA users from the JSC Spacecraft Software Engineering Branch as a VirtualBox® Image (.vdi file) or as a delivery of source and .gz files. Examples, users' guides, and other documentation are included in the delivery. These files have been transferred to DMTWG users at KSC and Glenn Research Center for analysis of problem reports.

The STAT tutorial with instructions for processing new datasets is included in Appendix B. The STAT user guide for installation and running examples of data processing in STAT and Flamenco+ is included in Appendix C. The user guide and abbreviated user guide for maintaining and updating the AO are included in Appendix D. The Flamenco+ tutorial for analyzing problem reports is included in Appendix E. The *User Guide for Trend Analysis with Flamenco+* is included in Appendix F.

11.0 Lessons Learned

No applicable lessons learned were identified for entry into the NASA Lessons Learned Information System.

12.0 Definition of Terms

Corrective Actions	Changes to design processes, work instructions, workmanship practices, training, inspections, tests, procedures, specifications, drawings, tools, equipment, facilities, resources, or material that result in preventing, minimizing, or limiting the potential for recurrence of a problem.
Finding	A conclusion based on facts established by the investigating authority.
Lessons Learned	Knowledge or understanding gained by experience. The experience may be positive, as in a successful test or mission, or negative, as in a mishap or failure. A lesson must be significant in that it has real or assumed impact on operations; valid in that it is factually and technically correct; and applicable in that it identifies a specific design, process, or decision that reduces or limits the potential for failures and mishaps, or reinforces a positive result.

Observation	A factor, event, or circumstance identified during the assessment that did not contribute to the problem, but if left uncorrected has the potential to cause a mishap, injury, or increase the severity should a mishap occur. Alternatively, an observation could be a positive acknowledgement of a Center/Program/Project/Organization's operational structure, tools, and/or support provided.
Ontology	Formal representation of the shared vocabulary and set of concepts that are used to name and describe entities in a domain, and the relationships between those concepts. The representation is usually includes a concept-sub-concept hierarchy. It can be used to reason about the domain entities.
Problem	The subject of the independent technical assessment.
Proximate Cause	The event(s) that occurred, including any condition(s) that existed immediately before the undesired outcome, directly resulted in its occurrence and, if eliminated or modified, would have prevented the undesired outcome.
Recommendation	An action identified by the NESC to correct a root cause or deficiency identified during the investigation. The recommendations may be used by the responsible Center/Program/Project/Organization in the preparation of a corrective action plan.
Root Cause	One of multiple factors (events, conditions, or organizational factors) that contributed to or created the proximate cause and subsequent undesired outcome and, if eliminated or modified, would have prevented the undesired outcome. Typically, multiple root causes contribute to an undesired outcome.

13.0 Acronyms List

AO	Aerospace Ontology
ATK	Alliant Techsystems, Inc.
ConOps	Concept of Operations
DMTWG	Data Mining and Trending Working Group
DR	Discrepancy Reports
Flamenco+	Flexible Information Access Using Metadata in Novel Combinations
JPL	Jet Propulsion Laboratory
JSC	Johnson Space Center
KSC	Kennedy Space Center
LaRC	Langley Research Center
MCR	Minimal Clausal Reconstruction

MEI Technologies®	Merging Excellence and Innovation Technologies®
MTSO	Management and Technical Support Office
NESC	NASA Engineering and Safety Center
NRB	NESC Review Board
NSC	NASA Safety Center
OSMS	Office of Safety and Mission Success
PI	Principal Investigator
S&K Aerospace	Salish and Kootenai Aerospace
STAT	Semantic Text Analysis Tool
UCF	University of Central Florida

14.0 References

1. N. Daraselia, A. Yuryev, S. Egorov, S. Novichkova, A. Nikitin, and I. Mazo, "Extracting Human Protein Interactions from MEDLINE using a Full-sentence Parser," *Bioinformatics,* 20 (5), 2004, pp. 604–611.
2. P. Hearst, "The Flamenco Search Interface Project," *http://flamenco.berkeley.edu/*.
3. J. Malin and D. Throop, "Basic Concepts and Distinctions for an Aerospace Ontology of Functions, Entities and Problems," *2007 IEEE Aerospace Conf. Proc*., CD-ROM, IEEE Aerospace, March 2007, pp. 1–18.
4. J. Malin, C. Millward, F. Gomez, and D. Throop, "Semantic Annotation of Aerospace Problem Reports to Support Text Mining," *IEEE Intelligent Systems*, 25 (5), September/October 2010, pp. 20–26.
5. J. Malin, C. Millward, H. Schwarz, F. Gomez, D. Throop, and C. Thronesbery, "Linguistic Text Mining for Problem Reports," *2009 IEEE Int'l Conf. Systems, Man and Cybernetics*, 2009, pp. 1578–1583.
6. H-M. Müller, E. Kenny, and P. Sternberg, "Textpresso: An Ontology-Based Information Retrieval and Extraction System for Biological Literature," *PLoS Biol.*, November 2004, 2(11), e309.
7. "Welcome to Protégé," *http://protege.stanford.edu/*.

15.0 Appendices

Appendix A. From the MCR to Semantic Interpretation: A Progress Report
Appendix B. STAT Tutorial
Appendix C. STAT User Guide
Appendix D. User Guide of Maintaining and Updating the Aerospace Ontology
Appendix E. Tutorial: Analyzing Problem Reports with Flamenco+
Appendix F. The Flamenco+ Faceted Browser User Guide for Trend Analysis

Appendix A. From the MCR to Semantic Interpretation: A Progress Report

From the MCR to Semantic Interpretation: A Progress Report

Fernando Gomez, Michael Gabilondo
Department of EECS
University of Central Florida, Orlando, FL 32816
gomez@eecs.ucf.edu, mgabilon@cs.ucf.edu

September 30, 2011

Contents

1 Introduction

Our research effort has been to investigate the identification of semantic roles and verb predicates from the output of the MCR. Our research has proceeded in two ways, developing new algorithms for semantic interpretation and implementing them in a semantic interpreter (SI). This report contains a detailed description of the SI, including installation instructions, a description of the tree TreeTagger and how to use it, MCR post-processor, the SI and examples of verb predicates and semantic roles for many sentences including those dealing with software requirements.

In addition, we have been investigating the following algorithms.

A) Algorithms that clean the output of the MCR before it is passed to the SI. Prior to activating the SI, the output of the MCR needs to be cleaned somewhat using the verb subcategorization information, in particular when the OBJi refers to a clause. The critical entries are obj and obj2, which in the output produced by the MCR may stand for time NPs, or subordinate clauses not subcategorized by the main verb. In a first approach, it was left to the SI (semantic interpreter) to make sense directly of the structures built by the MCR. However, these structures may contain too much noise (many OBJi) in long sentences for being handled directly by the SI without some cleaning. If the OBJi stands for an NP, it is left untouched, except for checking for time NPs, and it is up to the SI to make sense of it. However, if the OBji refers to a clause, the algorithm *Process OBJi That Stand for Clauses* is activated for each OBJi in the structure from left to right. There are two cases. Case a)If OBJi is equal to obj, obj2, or obj3 an algorithm is activated in order to determine if the OBJi is a complement phrase (CP). If the algorithm determines that the OBJi is a CP, it is left in the structure. If the OBJi is not a CP, it is erased from the structure, and the index of any OBJi still in the structure is replaced with OBJ(i-1). Case b) if the index of OBJi is greater than 3 (OBJi = obj4, or obj5, etc) and the OBJi refers to a infinitive, the algorithm assumes that the infinitive is a purpose infinitive. If the OBJi is not an infinitive, it is erased from the structure (not a CP). There are few outputs produced by the MCR having OBJi with an index equal or greater than 3.

B) Algorithms that improve the recognition of passive clauses by the MCR. There are frequent cases in which the MCR identifies a clause as passive when it is actually active, and vice-versa. In some cases, these errors are due to the parser which identifies wrongly a verb as VBD (past tense) when it is a VBN (past participle) and vice-versa. These errors cause the SI to miss a role, and in some cases a verb meaning. These algorithms use mostly syntactic principles based in the clause structure and verb subcategorization to repair the output of the MCR.

C) Algorithms that repair the MCR output when it wrongly identifies empty categories resulting from self-embedded relative clause. These algorithms use syntactic principles, verb and noun subcategorization to repair the MCR output. For instance, for the sentence "The belief that bats drink human blood is false," the parser output is:

"(S1 (S (NP (DT the) (NN belief) (SBAR (IN that) (S (NP (NNS bats)) (VP (VBP drink) (NP (JJ human) (NN blood)))))) (VP (AUX is) (ADJP (JJ false))) (. .)))"

and the MCR output is:

```
(G891
 (SUBJ ((DT the) (NOUN belief)) RELATIVE ((GENSYM G892 REL WHNP-M that)) VERB
  ((MAIN-VERB be is) (TENSE AUXVB)) SS (T) PRED ((ADJ false)))
 G892
 (SUBJ ((NOUN bats)) VERB ((MAIN-VERB drink drink) (TENSE VBP)) TYPE
  (REL WHNP-M that) SS (T) PARENT-VERB (be G891) OBJ ((ADJ human) (NOUN blood))
  MOVED-OBJ ((DT the) (NOUN belief)))))
```

The parser does not tell if the SBAR clause is a relative clause or not. The MCR assumes wrongly that the SBAR is a relative clause and incorrectly builds a MOVED-OBJ in the sentence for "drink." These algorithms will recognize that MOVED-OBJ is incorrect, and will delete it from the clause.

D) Algorithms that choose between subjects, when the MCR builds more than one subject for a clause. These algorithms choose between subjects based on the verb subcategorization and verb semantics. Potential subjects are entered by the MCR in the following order: first the subject of the main clause and then the object of the main clause if any. Thus, consider the sentence "She bought a book of history to learn the truth." The potential subjects of "learn" are "she" and "book." These algorithms would recognize "to learn" as a purpose infinitive, and select "she" as the subject of "learn." For the sentence, "The houses were bought to be sold" the potential subjects of "sold" are "unknown-agent" and "the houses," and these algorithms will select "houses" (main clause is passive) as the subject of "sold." For "He was told to be fair" the subjects entered for "be" are "unknown-agent" and "he." These algorithms will identify "to be" as an argument not as a purpose infinitive, and will select the second entry in the subject slot, namely"he," as the subject of "be." For the sentence "Huge ice blocks prevented him from going farther" the algorithms will select "him" as the subject of "going farther" based in the subcategorization of "prevent."

E) Algorithms that deal with "it" when it is a clausal substitute not a pronoun, i.e., "It was clear that acute rheumatic fever was a complication of group A streptococcal pharyngitis." In which the subject of "was clear" is "acute rheumatic fever was a complication of group A streptococcal pharyngitis" and "it" stands for the entire clause. Or, "It was/became obvious that the French had not seriously planned the attack." In which the subject of "was/became obvious" is "the French had not seriously planned the attack and "it" stands for the entire clause. There are many other examples, e.g., "It surprised her that the child ate," "It has been said that she saved the country," etc.

F) Algorithms that deal with temporal NPs , for instance "last night" in "Last night the children watch the puppet show," or "every day" in "They need to eat every day."

We will report on these algorithms and their testing in future progress reports.

2 Installation Instructions for the SI

The package includes the following top-level directories.

Directory	Description
common	Some shared functions
docs	Documentation, including this readme
mcr-postprocessor	MCR post-processor
mcr	Python module that interfaces with MCR
pos-tagger	Python module that interfaces with POS tagger
scripts	Includes the script to run the system
si	Semantic Interpreter
UCF_NLP-stanford-1.10	MCR that includes Stanford Parser

This package requires at least Python 2.7.x, but not Python 3. Treetagger and NLTK must be installed to run the system. The MCR, which is included, must also be installed and running prior to running the SI. Instructions are given below.

2.1 Install TreeTagger

Treetagger must be installed to the "treetagger" directory; in the top-level directory of this package, run "mkdir treetagger". The download and installation instructions are found at
http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/
Simplified instructions are also given below.

Download the following files to treetagger/

- Download Treetagger; for PC-Linux, the package is located at
  ftp://ftp.ims.uni-stuttgart.de/pub/corpora/tree-tagger-linux-3.2.tar.gz
- Download the tagging scripts,
  ftp://ftp.ims.uni-stuttgart.de/pub/corpora/tagger-scripts.tar.gz
- Download the English parameter file,
  ftp://ftp.ims.uni-stuttgart.de/pub/corpora/english-par-linux-3.1.bin.gz

Extract the tarballs; run the following commands in treetagger/.

- tar xfvz tree-tagger-linux-3.2.tar.gz
- tar xfvz tagger-scripts.tar.gz
- gunzip english-par-linux-3.1.bin.gz

2.2 Install NLTK

NLTK may be included as an optional package in your distribution. Alternatively, download and installation instructions may be found at http://www.nltk.org/.

NLTK comes with a large optional collection of data/corpora; only WordNet is required from this optional dataset.

2.3 Install and Run the MCR

This package includes the MCR with Stanford parser. The MCR requires SBCL 1.0.36, and may not work with newer versions of SBCL. The installation instructions for the MCR are found its README. Assuming all of the required perl and lisp modules have been installed, the following instructions should work.

From MCR directory (UCF_NLP-stanford-1.10/), run the following commands:

- rm data/asdf-registry/*
- make

The MCR does not daemonize, so it is useful to run it from a screen session.
To start MCR server, run the following commands:

- cd lisp
- ./setupsystem.sh

This MCR server is set up to only accept POS-tagged input by using a modified copy of java/stanford-parser/TCPServer.java. The original copy is found in TCPSever.java.ORIGINAL.

2.4 Test the system

The script scripts/run-si.sh may be used to run the entire SI pipeline for the plain-text input given in pos-tagger/input.txt. The output will be given in si/si-out. The script run-si.sh must be run from the scripts directory.

Run the command below to place the example sentence in the input file.

- echo "The apple was eaten by Mary with a fork." > pos-tagger/input.txt

While not required, the predicate below will allow the SI to assign semantic roles to the arguments of eat; if not specified, the SI will still output the grammatical relations of the verb. Place the following predicate definition in the file si/verb-predicates.

```
(eat
(verbs eat)
(human-agent (gr (subj)) (sr thing))
(theme (gr (obj-0)) (sr thing))
(instrumentality (gr (pp (prep with))) (sr thing)))
```

This predicate defines three arguments for the verb eat, which are the human-agent, theme, and instrumentality. The human-agent is realized by the subject, the theme is realized by the object, and the instrumentality is realized by a preposition headed by with.

If the sentence is passive, the SI automatically maps the grammatical subject to the object; as a result, the theme is "The apple". In passive sentences, the subject may be given by a noun phrase (NP) within a prepositional phrase (PP) headed by "by"; e.g., the NP "Mary" in the PP "by Mary" is the subject, which is the human-agent. The SI would output the same interpretation of the sentence was in active voice, as in "Mary ate the apple with the fork."

Once the MCR is running, from the root directory, you may run the following commands to interpret the example sentence:

- cd scripts
- ./run-si.sh

The script runs the following commands, which illustrates the SI pipeline:

```
cd ../pos-tagger/
./tagger.sh input.txt > output.txt

cd ../mcr/
./parser.py ../pos-tagger/output.txt > mcr-out

cd ../mcr-postprocessor/
./mcr.py ../mcr/mcr-out > mcr-postprocessor-out

cd ../si/
./si.py ../mcr-postprocessor/mcr-postprocessor-out > si-out
```

The output is given in si/si-out, but it is not easily readable since entire SI output is on one line. To run the SI in debugging mode and have it output to the terminal, run the following commands from the root directory:

- cd si/
- ./si_dev.py ../mcr-postprocessor/mcr-postprocessor-out

The debugging output is shown below.

```
SENT # 0

The apple was eaten by Mary with a fork .

(S1 (S (NP (DT The) (NN apple)) (VP (AUX was) (VP (VBN eaten) (PP (IN by)
(NP (NNP Mary))) (PP (IN with) (NP (DT a) (NN fork))))) (PERIOD .)))

(MCR
  (eaten-8
    (VERB
      (MAIN-VERB eaten eat)
      (VERB-TYPE VERB)
      (VOICE PASSIVE)
      (MODIFIERS ( (ID 6) ( (AUX was))))
      (TENSE VBN))
    (PP (ID 9) (PREP (IN by)) (NP (ID 12) ( (NNP Mary))))
    (PP (ID 13) (PREP (IN with)) (NP (ID 17) ( (DT a) (NN fork))))
    (POTENTIAL-SUBJECT-0 (ID 11) (NP (ID 12) ( (NNP Mary))))
    (SUBJECT-0 (ID 2) (NP (ID 4) ( (DT The) (NN apple))))))

(SI
  (eaten-8
    (pred verb-ont eat)
    (theme
      (np (id SUBJECT-0 4) (senses (thing (mod The) (head apple)))))
    (human-agent
      (np
        (id POTENTIAL-SUBJECT-0 12)
        (senses (thing (mod ) (head Mary)))))
    (instrumentality
      (pp
        (prep with)
        (np (id PP PP 1 17) (senses (thing (mod a) (head fork))))))))
```

3 How to use the SI (walk-through)

The SI pipeline includes four major components: the POS tagger (TreeTagger), the MCR (with Stanford Parser), the MCR post-processor and the SI. The input to the entire system is plain English text, and this text then passes through each of the major components in the pipeline. The final output of the SI is a list of semantically annotated clauses.

The user-defined verb predicates (verb meanings) and noun ontology determine the particular annotation for each clause. If a predicate has been defined for the verb of the clause, the system can determine the meaning of the main verb, the arguments and adjuncts of the verb labeled with semantic roles, and the senses of the head nouns of the arguments and adjuncts. If a predicate has not been defined, the system will still output the grammatical relations of each clause.

Consider the following example input sentence which is read from the input file pos-tagger/input.txt.

"This software module interfaces with a serial controller chip which is interfaced to a battery monitoring board ."

The following sections explain how this sentence is processed through each component of the pipeline.

3.1 POS Tagger

The pos-tagger module first tags the English input text using the POS tagger TreeTagger, and then applies a POST-tagger that can override tags in the TreeTagger output.

The POST-tagger is necessary because TreeTagger can make make tagging errors that do not allow for the correct interpretation of the sentence. For example, TreeTagger tags "interfaces" as a proper plural noun (NNS), when it is really a present third person singular verb, which has tag VBZ.

It is possible to write a context-dependent rule that tags "interfaces" with VBZ whenever it is succeeded by the preposition "with", which is an indicator that "interfaces" is being used as a verb. The pos-tagger contains the file pos-tagger/pos-words, which is a list of rules used by the POST-tagger for overriding tags in the output of the POS tagger (TreeTagger). The rule (interfaces VBZ (+1 with/IN)) specifies that "interfaces" should be assigned tag VBZ if the token that comes directly after it (indicated by +1) is "with" and has tag IN (see the section on pos-tagger/pos-words for details).

The output of the pos-tagger module for this sentence is shown below.

```
This/DT software/NN module/NN interfaces/VBZ with/IN a/DT serial/JJ
controller/NN chip/NN which/WDT is/VBZ interfaced/VBN to/TO a/DT battery/NN
monitoring/NN board/NN ./SENT
```

The script run-si.sh will run the POS tagger, but to run the POS tagger manually, you must be in the directory pos-tagger; the command

- ./tagger.sh input.txt > output.txt

will run the tagger on the English input text in input.txt and write the output tagged sentences to output.txt.

3.2 MCR

The POS tagged text (i.e., output of pos-tagger) becomes the input to the MCR. The MCR first parses the text using Stanford parser and constructs a set of clauses from the output of the parser; among other information, each clause includes the main verb, the voice of the clause (e.g., active or passive) and the grammatical relations, such as the subject, object and prepositional phrases. The output of Stanford parser for the example sentence is shown below.

```
(S1 (S (NP (DT This) (NN software) (NN module)) (VP (VBZ interfaces) (PP (IN
with) (NP-REL (NP (DT a) (JJ serial) (NN controller) (NN chip)) (SBAR (WHNP
```

```
(WDT which)) (S (VP (AUX is) (VP (VBN interfaced) (PP (TO to) (NP (DT a) (NN
battery) (NN monitoring) (NN board))))))))) (PERIOD .)))
```

The MCR builds a clause (which it calls a scope) for each of the main verbs in the sentence; in this case, clauses are built for "interfaces" and "interfaced". The actual output of the MCR is shown below, but it is not easily readable since it contains only indices into the parse tree.

```
[{"PREP-PHRASES": [8], "VERB": 7, "VOICE": "ACTIVE", "CONST-REP": 1, "SUBJECT":
[2]}, {"MODIFIERS": [21], "PREP-PHRASES": [24], "VERB": 23, "CLAUSE-TYPE":
"RELATIVE", "VOICE": "PASSIVE", "CONST-REP": 16, "SUBJECT": [11, 2]}]
```

A human-readable representation of this output is shown below, but it does not include all of the information of the actual output (e.g., modifiers are not listed).

```
SCOPE:  (VBZ interfaces) ACTIVE
PREP-PHRASES (ID 8): (PP (IN with) (NP-REL (NP (DT a) (JJ serial)
(NN controller) (NN chip)) (SBAR (WHNP (WDT which)) (S (VP (AUX is)
(VP (VBN interfaced) (PP (TO to) (NP (DT a) (NN battery) (NN monitoring)
(NN board))))))))

SUBJECT (ID 2): (NP (DT This) (NN software) (NN module))

SCOPE:  (VBN interfaced) PASSIVE
PREP-PHRASES (ID 24): (PP (TO to) (NP (DT a) (NN battery) (NN monitoring)
(NN board)))
SUBJECT (ID 11): (NP (DT a) (JJ serial) (NN controller) (NN chip))
SUBJECT (ID 2): (NP (DT This) (NN software) (NN module))
```

The script run-si.sh will run the MCR; however, to run the MCR manually, the MCR server must already be running (see Installation Instructions), and you must be in the directory mcr. The following command will run the MCR using the output of the pos-tagger as input, and will output to the file mcr-out.

- ./mcr.py ../pos-tagger/output.txt > mcr-out

3.3 MCR post-processor

The MCR post-processor transforms the output of the MCR; the transformation, among other things, includes detaching prepositional phrases from other constituents and performing verb stemming. The output of the MCR post-processor serves as input to the SI.

The MCR post-processor relies on WordNet as its lexicon to perform stemming. But, for those words not in WordNet, we have defined our own lexicon located in mcr-postprocessor/mcr-postprocessor-lexicon; before checking WordNet, the MCR post-processor first looks up words in the mcr-postprocessor-lexicon.

For example, the word "interface" is not a verb in WordNet, so we have defined our own entry in the mcr-postprocessor-lexicon, shown below.

```
(verb (root interface) (forms interfaces interfaced interfaced interfacing))
```

This entry means that 'interface' is a verb, and it lists four verb tenses in order of third-person-present, simple-past, past-participle and present-participle; the verb tenses must always be given in this order.

The output of the MCR post-processor is shown below.

```
(MCR
  (interfaced-23
    (VERB
      (MAIN-VERB interfaced interface)
      (VERB-TYPE VERB)
      (VOICE PASSIVE)
      (CLAUSE-TYPE RELATIVE)
      (MODIFIERS ( (ID 21) ( (AUX is))))
      (TENSE VBN))
    (PP
      (ID 24)
      (PREP (TO to))
      (NP (ID 30) ( (DT a) (NN battery) (NN monitoring) (NN board))))
    (SUBJECT-0
      (ID 11)
      (NP (ID 15) ( (DT a) (JJ serial) (NN controller) (NN chip))))
    (SUBJECT-1
      (ID 2)
      (NP (ID 5) ( (DT This) (NN software) (NN module))))
    (PARENT interfaces-7))
  (interfaces-7
    (VERB
      (MAIN-VERB interfaces interface)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (PP
      (ID 8)
      (PREP (IN with))
      (NP (ID 15) ( (DT a) (JJ serial) (NN controller) (NN chip))))
    (RELATIVE (ID 23) (CONJ (WDT which)) (CLAUSE interfaced-23))
    (SUBJECT-0
      (ID 2)
      (NP (ID 5) ( (DT This) (NN software) (NN module))))))
```

The MCR post-processor is run automatically by run-si.sh; however, to run the MCR post-processor manually, you must be in the directory mcr-postprocessor and you must run the command

- ./mcr.py ../mcr/mcr-out > mcr-postprocessor-out

3.4 Semantic Interpreter

The input to the Semantic Interpreter is the output of the MCR post-processor. The SI relies on user-input definitions of verb and noun meanings. The verb meanings are called verb predicates, and they are defined in si/verb-predicates.

We have defined two predicates for the verb "interface" for a certain domain. The most common meaning means to connect two components, as in the sentence, "This software module interfaces with a serial controller chip"; the two components being connected are "this software module" and "a serial controller chip", which our predicate names the theme and co-theme, respectively. The following predicate is sufficient to identify the two arguments in our example sentence.

```
(interface-connect
(verbs interface)
(theme (gr (subj-if-not-obj-0)) (sr thing))
(co-theme (gr (pp (prep with))) (sr thing)))
```

The name of the predicate is interface-connect, and the entry "(verbs interface)" specifies the verbs which may mean this predicate; more than one verb may be included in by separating them with spaces.

The predicate specifies two arguments, which it labels the theme and co-theme. In order for a grammatical relation to be labeled as the theme argument, it must satisfy the restrictions in the (gr) and (sr) entries of the argument. The (gr) entry specifies the grammatical relations which may map to this argument, and the (sr) entry (which stands for selectional restriction) restricts the sense of the head noun of the NP of the argument, if applicable; specifying thing as the sr is equivalent to having no restriction on the head noun sense.

The theme specifies the gr (subj-if-not-obj-0), which means the grammatical relation is true if the clause has a subject but not an object. The co-theme specifies the gr (pp (prep with)), which means the grammatical relation is true if the clause contains a post-verbal prepositional phrase headed by the preposition with.

However, the definition of this predicate is not sufficient to capture all uses of this meaning of interface. For example, the verb may be used in passive voice, as in "This software module is interfaced with a serial controller chip". In this case, "This software module" is the object and not the subject, so subj-if-not-obj-0 will not match "This software module". This can be easily solved by adding the grammatical relation "passive-subj" to the list of grammatical relations specified in the theme. The SI also maps passive subjects to objects, so it is possible to write (obj-0) instead of (passive-subj). The revised definition is shown below. We have also added another preposition "to" to the co-theme, since it is possible to use "to" instead of "with", in this case.

```
(interface-connect
(verbs interface)
(theme (gr (subj-if-not-obj-0) (passive-subj)) (sr thing))
(co-theme (gr (pp (prep with to))) (sr thing)))
```

The output of the SI with only this predicate defined is shown below.

```
(SI
  (interfaced-23
    (pred verb-ont interface-connect)
    (theme
      (np
        (id SUBJECT-0 15)
        (senses (thing (mod a serial controller) (head chip)))))
    (co-theme
      (pp
        (prep to)
        (np
          (id PP PP 0 30)
          (senses (thing (mod a battery monitoring) (head board)))))))
  (interfaces-7
    (pred verb-ont interface-connect)
    (theme
      (np
        (id SUBJECT-0 5)
        (senses (thing (mod This software) (head module)))))
    (co-theme
      (pp
        (prep with)
        (np
          (np
            (id PP PP 0 15)
```

```
        (senses (thing (mod a serial controller) (head chip))))
      (rel
        (id RELATIVE PP-8 0 23)
        (conj which)
        (clause interfaced-23)))))))
```

In this domain, there is another sense for "interface" which means to transfer something from one location to another. An example of this usage is given in the sentence below.

"The CIU interfaces crew audio and biomed data to/from the Vehicle Control Network ."

We may define a separate predicate to capture this meaning, which is shown below, along with the output of the SI for the sentence above.

```
(interface-transfer
(verbs interface)
(inanimate-cause (gr (subj)) (sr thing))
(theme (gr (obj-0)) (sr thing))
(goal (gr (pp (prep to))) (sr thing))
(source (gr (pp (prep from))) (sr thing)))

(SI
  (interfaces-6
    (pred verb-ont interface-transfer)
    (source
      (pp
        (prep from)
        (np
          (id PP PP 1 21)
          (senses (thing (mod the Vehicle Control) (head Network))))))
    (theme
      (np
        (AND
          (np
            (id OBJECTS-0 10)
            (senses (thing (mod crew) (head audio))))
          (np
            (id OBJECTS-0 14)
            (senses (thing (mod biomed) (head data)))))))
    (goal
      (pp
        (prep to)
        (np
          (id PP PP 0 21)
          (senses (thing (mod the Vehicle Control) (head Network))))))
    (inanimate-cause
      (np (id SUBJECT-0 4) (senses (thing (mod The) (head CIU)))))))
```

The SI will automatically select the correct predicate for each clause. To choose the predicate for the clause, the SI tries to match the roles for each predicate, and chooses the predicate with the maximum number of roles satisfied as the meaning of the clause. If two or more predicates have the same maximum number of roles satisfied, then all such predicates are outputted as candidate meanings. It is possible to specify a priority for each predicate. The SI will select the predicate with the highest priority.

Consider again the first example sentence for which we defined the predicate interface-connect.

"This software module interfaces with a serial controller chip which is interfaced to a battery monitoring board ."

As we saw previously, the predicate interface-connect is taken as the meaning for both of these usages of interface. However, we have introduced an additional predicate interface-transfer, which is also chosen as a meaning for "interfaced", as shown below.

```
(interfaced-23
  (pred verb-ont interface-transfer)
  (theme
    (np
      (id SUBJECT-0 15)
      (senses (thing (mod a serial controller) (head chip)))))
  (goal
    (pp
      (prep to)
      (np
        (id PP PP 0 30)
        (senses (thing (mod a battery monitoring) (head board)))))))

(interfaced-23
  (pred verb-ont interface-connect)
  (theme
    (np
      (id SUBJECT-0 15)
      (senses (thing (mod a serial controller) (head chip)))))
  (cotheme
    (pp
      (prep to)
      (np
        (id PP PP 0 30)
        (senses (thing (mod a battery monitoring) (head board)))))))
```

Both of these meanings appear to be correct. For interface-transfer, "a serial controller chip" is transfered to "a battery monitoring board"; for interface-connect, "a serial controller chip" connects to "a battery monitoring board". However, it is also possible that the SI outputs two different meanings for a verb, one or more of which may be incorrect.

Since interface-connect occurs more frequently in our domain, we may specify that this meaning gets chosen whenever there is a tie. The predicate definitions allow a (priority i) entry, where i is an integer; the predicate with the lowest value of i is chosen as the meaning of the verb in case of ties. If a priority is not specified, i has the highest possible value (i.e., lowest priority). All tied predicates are output in sorted order by priority.

We may modify the predicate interface-connect by adding (priority 1) to prefer this meaning over interface-transfer whenever there is a tie. The modified predicate is shown below.

```
(interface-connect
(verbs interface)
(priority 1)
(theme (gr (subj-if-not-obj-0) (passive-subj)) (sr thing))
(cotheme (gr (pp (prep with to))) (sr thing)))
```

It is possible to specify concepts other than thing in the selectional restrictions, but we did not utilize the noun ontology in this example. For an example with the noun ontology, see Section "Detailed Description of Verb Predicates for an Example Application".

4 System Configuration

4.1 pos-tagger/input.txt

This is the file which run-si.sh expects as an input file containing plain English text.

The text in the input file may be in one of these two formats (or a combination of the two):

- Each sentence is on one line by itself, untokenized and ending with punctuation.
- The text is tokenized, but no sentence splitting has been performed.

Each resulting tagged output sentence is assigned an ID in the form "SENT # i", which is carried along to each module and which will be present in the output of the SI.

4.2 pos-tagger/pos-words

These rules are applied to the POS-tagged output of Treetagger to modify tags. Each rule specifies the surface word form (without stemming), the new tag, and a list of context-dependent rules which must be satisfied to change the tag.

The tags should be in Penn Treebank format, which are used by Stanford parser:
http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Treetagger uses its own tagset, but those tags are automatically converted to Penn Treebank tags prior to running the tag changer. The Treetagger tagset is found at
http://courses.washington.edu/hypertxt/csar-v02/penntable.html

A sample input file is given below.

```
(and/or CC)
(purge NN)
(to/from IN)
(interfaces VBZ (-1 NNP NNPS))
(interfaces VBZ (+1 with/IN))
(power-up VB)
```

The first rule is (and/or CC), which changes every occurrence of "and/or" to CC in the output of the POS tagger. The tag CC stands for coordinating conjunction, and it is the tag assigned to words like "and" and "or", but TreeTagger assigns an incorrect tag to "and/or". Similarly, the rule (purge NN) changes the tag of every occurrence of the word "purge" to NN. So, "purge" becomes a noun always.

These rules are useful to correct the output of Treetagger for specific corpora. For example, if "purge" is a noun every time in the target corpora, but Treetagger often marks it as a verb, the rule (purge NN) is sufficient to fix the tag. If the word appears both as a noun and a verb, then it is necessary to write a context-dependent rule.

For each rule, any number of contexts can be added; e.g, the first rule for the target word "interfaces" specifies context (-1 NNP NNPS), and the second rule specifies context (+1 with/IN). The context (-1 NNP NNPS) means that the target word should have its tag changed only if the token directly to the left of the target word has tag NNP or NNPS.

In order for a rule to override a tag in the POS tagger output, all of the rule's context entries must match. Each context specifies an offset from the target word (e.g., interfaces, in this case), and a list of tags, one of which must match the offset word. The offset is a + or - sign followed by an integer; e.g., -1 means one word to the left of the target word, but in general any integer may be specified. The rest of the entries in the context are POS tags either in the form POS (e.g., NNP) or word/POS (e.g., with/IN); in order for the context entry to match, the offset word must match one of the POS or word/POS entries in the context. If all context entries match, then the tag for the target word will be changed to the specified tag.

Each line <entry> has the following form:

```
<entry> := (WORD NEW-TAG <context-rule>*)
<context-rule> := (<offset> <tag-list>)
<offset> := + or - followed by an integer, e.g., -2
<tag-list> := [word/]tag [word/]tag [word/]tag ...
```

4.3 mcr-postprocessor/mcr-postprocessor-lexicon

The system uses WordNet as its lexicon. But, for those words not in WordNet, we have defined our own lexicon called mcr-postprocessor-lexicon. The system relies on WordNet as its lexicon to perform stemming, but it first looks up words in the mcr-postprocessor-lexicon.

A sample entry is given below:

```
(verb (root interface) (forms interfaces interfaced interfaced interfacing))
```

This entry means that 'interface' is a verb, and it lists four verb tenses in order of third-person-present, simple-past, past-participle and present-participle; the verb tenses must always be given in this order. Other example entries are given below.

```
(verb (root stem-form) (forms third-person-present simple-past past-participle
present-participle))

(verb (root buy) (forms buys bought bought buying))
(verb (root eat) (forms eats ate eaten eating))
(verb (root have) (forms has had had having))
(verb (root become) (forms becomes became become becoming))
```

Currently, this lexicon is only used by the MCR post-processor to stem verbs, and not in the SI; this will be changed in the future.

4.4 si/noun-ontology

Each entry in the noun ontology specifies a noun concept (sense).

For example, the two entries below define two senses for the word "batter" (batter.n.01 and batter.n.02) and one sense for "hitter", "slugger" and "batsman" (batter.n.01). batter.n.01 also defines a hypernym ballplayer.n.01 and batter.n.02 defines a hypernym concoction.n.01.

```
(batter.n.01
  (nouns batter hitter slugger batsman)
  (parents ballplayer.n.01))
(batter.n.02 (nouns batter) (parents concoction.n.01))
```

4.5 si/verb-predicates

Each verb predicate entry (verb meaning) may specify the arguments and adjuncts of the predicate, the list of verbs which may be used to mean that predicate, and a list of superpredicates from which arguments and adjuncts are inherited. Each argument or adjunct definition specifies its semantic role (i.e., a label with which to identify that argument), along with grammatical restrictions (gr) and semantic restrictions (sr).

The general form for a predicate definition is given below; * indicates that the preceding entry may appear zero or more times, and ? indicates that the preceding entry is optional.

```
(PREDICATE-NAME
(verbs LIST-OF-VERBS)?
(adjuncts LIST-OF-ADJUNT-NAMES)?
(ROLE-NAME (gr LIST-OF-GR) (sr LIST-OF-SR))*
(parents LIST-OF-PREDICATES)?)
```

PREDICATE-NAME is the name given to the predicate. LIST-OF-VERBS is a space separated list of verb names, and multiword verbs have their words separated by underscores. The entry (adjuncts LIST-OF-ADJUNT-NAMES) specifies roles in this predicate that are adjuncts instead of arguments, and LIST-OF-ADJUNCT-NAMES is a space separated list of role names which are to be treated as adjuncts instead of arguments. The entry (ROLE-NAME (gr LIST-OF-GR) (sr LIST-OF-SR)) defines an argument or adjunct for the predicate. LIST-OF-SR is space separated list of concepts from the noun ontology, and LIST-OF-GR is a space separated list of grammatical relations.

The list of possible grammatical relations is given below; LIST-OF-PREPS is a space separated list of prepositions.

GR	(adjp)
Description	Post-verbal adjective phrase
Example	The valve failed [open]
GR	(advp)
Description	Post-verbal adverb phrase
Example	He went [south]
GR	(appos)
Description	Post-nominalization apposition (nominalization predicates only)
Example	The receptor, [p1] binds to p2
GR	(cp)
Description	Post-verbal complement
Example	He knew [Mary told the truth]
GR	(cp-inf)
Description	Post-verbal complement introduced by infinitive (to)
Example	The valve fails [to open]
GR	(cp-prep (prep LIST-OF-PREPS))
Description	Post-verbal complement introduced by a preposition from LIST-OF-PREPS
Example	He gained money [from selling his goods to Mary]
GR	(cp-that)
Description	Post-verbal complement introduced by 'that'
Example	He knew [that Mary told the truth]
GR	(mod)
Description	Modifier of nominalization (nominalization predicates only)
Example	It allows [heat] transfer from one component to the other
GR	(obj-0)
Description	First post-verbal NP if active, grammatical subject if passive
Example	She gave [John] cake
GR	(obj-1)
Description	Second post-verbal NP
Example	She gave John [cake]
GR	(passive-subj)
Description	Grammatical subject, if the clause is in passive voice
Example	[The valve] was interfaced with the component
GR	(pp (prep LIST-OF-PREPS))
Description	Post-verbal PP headed with a preposition from LIST-OF-PREPS
Example	Mary ate the apple [with the fork]
GR	(subj)
Description	Grammatical subject if active voice, direct object if passive
Example	[Peter] read the book
GR	(subj-if-not-obj-0)
Description	Subject if the clause does not have a post-verbal NP
Example	[The protein] interacts with the molecule
GR	(subj-if-obj-0)
Description	Subject if the clause has a post-verbal NP
Example	[The wind] broke the door

5 Detailed Description of Verb Predicates for an Example Application

Suppose we want to extract information from news articles about violent attacks. We may begin by building a predicate (verb sense) for the verb fire meaning to use a gun to shoot bullets or bombs at someone.

In general, the verb fire is ambiguous, and has 8 senses in Longman Dictionary of Contemporary

English (LDOCE) (http://www.ldoceonline.com/dictionary/fire_2).

For the sense we are interested in, LDOCE lists the following subcategorization frames:

1. fire at/on/into; e.g., "Soldiers fired on the crowd."
2. fire something at somebody; e.g., "The police fired two shots at the suspects before they surrendered."
3. fire a gun/weapon/rifle etc (=make it shoot);, e.g., "the sound of a gun being fired."
4. fire bullets/missiles/rockets etc; e.g., "Guerrillas fired five rockets at the capital yesterday, killing 23 people."

The following predicate captures these syntactic variations.

```
(fire-projectiles
(verbs fire)
(human-agent (gr (subj)) (sr human social-group))
(instrumentality (gr (obj-0)) (sr weapon))
(theme (gr (obj-0)) (sr projectile))
(goal (gr (pp (prep at on into))) (sr thing)))
```

The entry (verbs fire) specifies the list of verbs that may mean fire-projectiles; verbs are separated by spaces and multiword verbs have their words separated by _, like in give_up. The following entry specifies an argument of the predicate with semantic role human-agent.

```
(human-agent (gr (subj)) (sr human social-group))
```

The role human-agent represents the human or social-group that causes the action, e.g., the soldiers or police.

The argument definition also specifies syntactic and semantic constraints. The syntactic constraints are specified by the (gr) entry, which is a list of grammatical relations that may realize that argument. For example, the human-agent argument specifies the (subj) grammatical relation, which means that this argument may only appear as the subject of the verb fire. More than one grammatical relation may be specified in the (gr) list, and they will be tried in order.

The semantic constraints come in the form of selectional restrictions specified by the (sr) entry. Each entry in the (sr) list is a concept from the noun ontology; for a constituent (e.g., an NP or PP) to match an argument, the head noun of the NP of the argument must have a sense in the noun ontology that is subsumed by one of the concepts in the selectional restrictions list.

If a selectional restriction is not desired, 'thing' may be specified as the concept and any head noun will satisfy the selectional restriction, even nouns that are not in the ontology. Therefore, it may be convenient to organize the ontology and set 'thing' as the root node; the root concept may be changed to something other than 'thing' in si/si_conf.py. Also, not all grammatical relations involve NPs, e.g., (cp); in these cases, (sr thing) should be specified.

Returning to the examples for the verb fire, the human-agent argument would match "Soldiers" in (1), "The police" in (2) and "Guerrillas" in (4); (3) has no subject, so the human-agent argument would not be realized.

The object of 'fire' may be a weapon, as in example (3) (i.e., "a gun being fired"); note that this clause is passive, so the grammatical subject is taken as the object. The object may also be a projectile, as in example (4) (i.e., "fire five rockets"). The object has a different semantic role depending on whether the head noun is a weapon or a projectile.

If the object is a weapon, then the semantic role of the argument is the instrumentality, since it is the instrument used to fire shots at someone. If the object is a projectile, then the semantic role is the theme, the thing that suffers the action. The final argument is the goal, i.e., where the shots (the theme) were fired.

Other verbs may also be used to mean the same thing as fire-projectile, e.g., the verb shoot. The sense of shoot in LDOCE that corresponds most closely to fire-projectiles is sense meaning to make a bullet or arrow come from a weapon. The following example sentences from LDOCE demonstrate the subcategorization frames for this sense of shoot.

1. "Two guys walked in and started shooting at people."
2. "The soldiers had orders to shoot to kill."
3. "They shot arrows from behind the thick bushes."
4. "Tod's grandfather taught him to shoot a rifle."

In (1), "at people" is an argument with semantic role goal, and note that this role has the same meaning as the goal in fire-projectile; the subject, "Two guys", is an argument with role human-agent. The goal may be defined as (goal (gr (pp (prep at))) (sr thing)) and the human-agent may be defined as (human-agent (gr (subj)) (sr human organization)).

In (2), "to kill" has the purpose role, and this argument may be defined as (purpose (gr (cp-inf)) (sr thing)). In (3), "arrows" is the theme (i.e., the projectile being shot), and "from behind the thick bushes" is an adjunct with semantic role at-loc (i.e., where the action is taking place). Adjuncts are different from arguments in that they are not specific to the verb meaning of interest, but may appear in general with other verbs. In most cases, the purpose is also an adjunct, but it in this case it is an argument (i.e., in "shoot to kill"). In (4), "a rifle" is the instrumentality, (i.e., the weapon used to shoot projectiles).

The predicate may be defined as follows.

```
(shoot-projectiles
(verbs shoot)
(human-agent (gr (subj)) (sr human social-group))
(instrumentality (gr (obj-0)) (sr weapon))
(theme (gr (obj-0)) (sr projectile))
(purpose (gr (cp-inf)) (sr thing))
(goal (gr (pp (prep at))) (sr thing)))
```

Predicates also allow definition of superpredicates, i.e., more general verb meanings. The predicate inherits its arguments from superpredicates but may also add new arguments or extend the restrictions of an inherited argument by defining an argument with the same name. The following defines a more general predicate fire-shoot, and two subpredicates shoot-projectiles and fire-projectiles. The adjunct at-loc is also defined in the root action predicate.

```
(action
(adjuncts at-loc)
(at-loc (gr (pp (prep from-behind)) (sr thing))))

(fire-shoot
(human-agent (gr (subj)) (sr human social-group))
(instrumentality (gr (obj-0)) (sr weapon))
(theme (gr (obj-0)) (sr projectile))
(parents action))

(shoot-projectiles
(verbs shoot)
(purpose (gr (cp-inf)) (sr thing))
(goal (gr (pp (prep at))) (sr thing))
(parents fire-shoot))
```

```
(fire-projectiles
(verbs fire)
(goal (gr (pp (prep at on into))) (sr thing))
(parents fire-shoot))
```

Other senses of the verb fire may also be present in the news articles, such as the sense meaning to terminate the employment of someone. If this is the case, predicates for the other senses should be defined; otherwise, all of the actual different senses will incorrectly map to the same predicate.

If a predicate is defined for fire-terminate-employment, whenever fire occurs, the SI will try both fire-terminate-employment and fire-projectiles, and choose the predicate with the largest number of matching roles as the meaning of the verb. The selectional restrictions of the arguments also help prevent picking an incorrect predicate.

The concepts that appear in the selectional restrictions are defined in the noun ontology, i.e., thing, human, social-group, weapon and projectile. We must define senses for all head nouns of arguments; however, if (sr thing) is used, the head noun does not have to be in the ontology in order to satisfy the selectional restriction, since all head nouns not in the ontology are assigned the sense thing. However, thing must be an entry in the ontology.

Below we define senses for only the example sentences above. Currently, pronouns like they and him are not handled explicitly, so we treat them as nouns in the definitions below.

```
(thing )
(human (nouns soldier guerrilla guy they grandfather) (parents thing))
(social-group (nouns police) (parents thing))
(projectile (nouns shot rocket arrow) (parents thing))
(weapon (nouns gun rifle) (parents thing))
```

The output of the SI for one of the example sentences is shown below.

"The police fired two shots at the suspects before they surrendered ."

```
(SI
  (surrendered-21
    (pred verb-ont nil)
    (subj (np (id SUBJECT-0 19) (senses (thing (mod ) (head they))))))
  (fired-6
    (pred verb-ont fire-projectiles)
    (theme
      (np
        (id OBJECTS-0 9)
        (senses (projectile (mod two) (head shots)))))
    (human-agent
      (np
        (id SUBJECT-0 4)
        (senses (social-group (mod The) (head police)))))
    (goal
      (pp
        (prep at)
        (np
          (id PP PP 0 14)
          (senses (thing (mod the) (head suspects))))))
    (cp-0
      (rel (id OBJECTS-1 21) (conj before) (clause surrendered-21)))))
```

6 Nominalizations (Advanced Topic)

Verb nominalizations are verb-like nouns that have clause structure. For example, the usage of interaction in "the interaction of A with B" is a nominalization that comes from the verb interact. The nominalization has two arguments, "of A" and "with B".

The MCR post-processor can output clauses for nominalizations as well as for verbs. The file **mcr/noms** contains a single verb on each line; clauses will be created for any head noun that is derived from one of these verbs. In order to find the verb form for nominalizations, a list of rules is given in **mcr/nom-suffixes**. An example rule is (SUFFIX ation e), which is read as, "if the suffix is ation, replace ation with e, and check if the resulting string is a verb in WordNet". The SI uses a separate ontology of predicates for nominalizations, which is located in **si/nom-predicates**. Predicates defined in this file will only only apply to nominalization clauses in the MCR post-processor.

There are also other nouns, which are not nominalizations but subcategorize for prepositions; entries in the noun ontology support subcategorization information which influences the attachment decisions of the SI. For example, if ligand subcategorizes for the preposition 'for', this information may be specified with a (subcat) entry as shown below.

```
(ligand
(nouns ligand)
(subcat (prep for) (sr thing))
(parents interaction-property))
```

7 Appendix

The appendix contains predicates we have defined for a particular domain and example output of the SI for selected sentences in that domain.

7.1 Example Predicates

```
(action
(adjuncts purpose)
; load SM CMD for post sep ops
(purpose (gr (pp (prep for))) (sr thing))
(manner (gr (cp-prep (prep through))) (sr thing)))

(state )

; Electrical Device Fails Open .
(fail-state
(verbs fail)
(theme (gr (subj)) (sr thing))
(at-state (gr (adjp)) (sr thing))
(parents action))

; PTT switch on the CIU fails to transmit audio due to internal failure of the switch .
(fail-action
(verbs fail)
(inanimate-cause (gr (subj)) (sr thing))
(theme (gr (cp-inf)) (sr thing))
(parents action))

; the Avionics Software shall fire the pyro isolation valve to enable GN2 flow .
(fire-activate
(verbs fire)
```

```
(inanimate-cause (gr (subj)) (sr thing))
(theme (gr (obj-0)) (sr thing))
(parents action))


; This software module interfaces with a serial controller chip which is interfaced to a batte
(interface-connect
(verbs interface)
(priority 1)
(theme (gr (subj-if-not-obj-0) (passive-subj)) (sr thing))
(cotheme (gr (pp (prep with to))) (sr thing))
(parents action))

; The CIU interfaces crew audio and biomed data to/from the Vehicle Control Network .
(interface-transfer
(verbs interface)
(inanimate-cause (gr (subj)) (sr thing))
(theme (gr (obj-0)) (sr thing))
(goal (gr (pp (prep to))) (sr thing))
(source (gr (pp (prep from))) (sr thing))
(parents action))

(provide-transfer
(verbs provide)
(inanimate-cause (gr (subj)) (sr thing))
(theme (gr (obj-0)) (sr thing))
(goal (gr (pp (prep to))) (sr thing))
(parents action))

;Launch vehicle stack-up is loaded axially by the jettison motor.
; (THEME, what is being loaded, is missing; assuming stack-up is the goal)
; (are we sure stack-up is the goal and not the THEME?)

;Avionics capability for GSE interfacing functions for SW loading, test sep
; affected if SM CMDs not loaded for post sep ops.
;  ("loaded" is passive, but the MCR marks it as ACTIVE, so obj-0 will not match SM CMD.
;   "load SM CMD for post sep ops".)
;  (it is unclear if SM CMD is the theme or goal, since we don't know what SM CMD is)
(load-put
(verbs load)
(inanimate-cause (gr (subj)) (sr thing))
(goal (gr (obj-0)) (sr thing))
(manner (gr (advp)) (sr thing))
(parents action))



; All flight sw (C&DH, PM&D,GNC, D&C, C&T, ADL, MM, SM, ECLSS) applications are
; hosted on the shared resources of the VMC.  ;   (host applications on shared
; resources => shared resources host applications) ; The SCCA's main controller
; card hosts higher layer functions
(host-contain
(verbs host)
```

```
(theme (gr (obj-0)) (sr thing))
(at-loc (gr (subj) (pp (prep on))) (sr thing))
(parents state))

; Prefer to replace unit to maintain quick crew notification of urgent situations.
(maintain
(verbs maintain)
(theme (gr (obj-0)) (sr thing))
(parents action))


;Micro-RIU function is to collect and format vehicle structure sensor readings
;into digital data for transfer to memory storage in the VMC.
(collect
(verbs collect)
(inanimate-cause (gr (subj)) (sr thing))
(theme (gr (obj-0)) (sr thing))
(parents action))

(format-change
(verbs format)
(inanimate-cause (gr (subj)) (sr thing))
(theme (gr (obj-0)) (sr thing))
(to-state (gr (pp (prep into))) (sr thing))
(parents action))

; This heat exchanger allows heat rejection from the internal Dowfrost HD loop
; to the external refrigerant loop
(allow-transfer
(verbs allow)
(inanimate-cause (gr (subj)) (sr thing))
(theme (gr (obj-0)) (sr thing))
(parents action))


; Figure 3 describes the SAFER FU subsystems which include the Hand Controller
; Module (HCM) and the Propulsion Module (PM)
(include
(verbs include)
(thing-described (gr (subj)) (sr thing))
(attribute-described (gr (obj-0) (cp)) (sr thing))
(parents state))


; Tri-Band Patch Low Gain Antenna  Transmits and receives S-Band data to/from
; the Space Network
; =>
; antenna transmits s-band data to the space network
; antenna receives s-band data from the space network
(transmit-transfer
(verbs transmit)
(inanimate-cause (gr (subj)) (sr thing))
(theme (gr (obj-0)) (sr thing))
```

```
(goal (gr (pp (prep to))) (sr thing)))

(receive-transfer
(verbs receive)
(inanimate-cause (gr (subj)) (sr thing))
(theme (gr (obj-0)) (sr thing))
(source (gr (pp (prep from))) (sr thing)))

; A steady-state actuated valve in between the high pressure nitrogen tank in
; the CM and the cabin controls supply of fire suppressive nitrogen to the
; Avionics and ECLSS subsystem bays
(control-manipulate
(verbs control)
(inanimate-cause (gr (subj)) (sr thing))
(theme (gr (obj-0) (cp)) (sr thing))
(parents action))
```

7.2 Example Output

This section contains example output of the SI for selected sentences. Each sentence contains the output of Stanford Parser, the MCR post-processor and the SI.

```
SENT # 4

EPIC/GVSC data or control signal fails high or low .

(S1 (S (NP-COORD (NP (NN EPIC/GVSC) (NNS data)) (CC or) (NP (NN control)
(NN signal))) (VP (VBZ fails) (ADJP-COORD (JJ high) (CC or) (JJ low)))
(PERIOD .)))

(MCR
  (fails-11
    (VERB
      (MAIN-VERB fails fail)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0
      (ID 12)
      (ADJP
        (ID 12)
        (
          (OR (ADJP (ID 13) ( (JJ high))) (ADJP (ID 15) ( (JJ low)))))))
    (SUBJECT-0
      (ID 2)
      (NP
        (ID 2)
        (
          (OR
            (NP (ID 5) ( (NN EPIC/GVSC) (NNS data)))
            (NP (ID 9) ( (NN control) (NN signal)))))))))
```

```
(SI
  (fails-11
    (pred verb-ont fail-state)
    (theme
      (np
        (OR
          (np
            (id SUBJECT-0 5)
            (senses (thing (mod EPIC/GVSC) (head data))))
          (np
            (id SUBJECT-0 9)
            (senses (thing (mod control) (head signal)))))))
    (at-state
      (adjp
        (OR
          (adjp (id OBJECTS-0 13) (words high))
          (adjp (id OBJECTS-0 15) (words low)))))))

SENT # 5

DU fails to power-up after power is cycled .

(S1 (S (NP (NNP DU)) (VP (VBZ fails) (S-INF (VP (TO to) (VP (VB power-up) (SBAR
(IN after) (S (NP (NN power)) (VP (AUX is) (VP (VBN cycled)))))))))
(PERIOD .)))

(MCR
  (cycled-19
    (VERB
      (MAIN-VERB cycled cycle)
      (VERB-TYPE VERB)
      (VOICE PASSIVE)
      (MODIFIERS ( (ID 17) ( (AUX is))))
      (TENSE VBN))
    (SUBJECT-0 (ID 14) (NP (ID 15) ( (NN power))))
    (PARENT power-up-10))
  (fails-5
    (VERB
      (MAIN-VERB fails fail)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0
      (ID 6)
      (RELATIVE (ID 10) (CONJ (TO to)) (CLAUSE power-up-10)))
    (SUBJECT-0 (ID 2) (NP (ID 3) ( (NNP DU)))))
  (power-up-10
    (VERB
      (MAIN-VERB power-up power-up)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 8) ( (TO to))))
      (TENSE VB))
```

```
    (OBJECTS-0
      (ID 11)
      (RELATIVE (ID 19) (CONJ (IN after)) (CLAUSE cycled-19)))
    (SUBJECT-0 (ID 2) (NP (ID 3) ( (NNP DU))))
    (PARENT fails-5)))

(SI
  (cycled-19
    (pred verb-ont nil)
    (obj-0
      (np (id SUBJECT-0 15) (senses (thing (mod ) (head power))))))
  (fails-5
    (pred verb-ont fail-action)
    (theme (rel (id OBJECTS-0 10) (conj to) (clause power-up-10)))
    (inanimate-cause
      (np (id SUBJECT-0 3) (senses (thing (mod ) (head DU))))))
  (power-up-10
    (pred verb-ont nil)
    (cp-0 (rel (id OBJECTS-0 19) (conj after) (clause cycled-19)))
    (subj (np (id SUBJECT-0 3) (senses (thing (mod ) (head DU)))))))

SENT # 6

PTT switch on the CIU fails to transmit audio due to internal failure of the switch .

(S1 (S (NP (NP (NNP PTT) (NN switch)) (PP (IN on) (NP (DT the) (NNP CIU))))
(VP (VBZ fails) (S-INF (VP (TO to) (VP (VB transmit) (NP (NN audio)) (PP (JJ
due) (TO to) (NP (NP (JJ internal) (NN failure)) (PP (IN of) (NP (DT the) (NN
switch)))))))) (PERIOD .)))

(MCR
  (fails-12
    (VERB
      (MAIN-VERB fails fail)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0
      (ID 13)
      (RELATIVE (ID 17) (CONJ (TO to)) (CLAUSE transmit-17)))
    (SUBJECT-0 (ID 2) (NP (ID 5) ( (NNP PTT) (NN switch))))
    (PP (ID 6) (PREP (IN on)) (NP (ID 10) ( (DT the) (NNP CIU)))))
  (transmit-17
    (VERB
      (MAIN-VERB transmit transmit)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 15) ( (TO to))))
      (TENSE VB))
    (PP
      (ID 20)
      (PREP (JJ due) (TO to))
      (NP (ID 26) ( (JJ internal) (NN failure))))
```

```
    (PP (ID 27) (PREP (IN of)) (NP (ID 31) ( (DT the) (NN switch))))
    (OBJECTS-0 (ID 18) (NP (ID 19) ( (NN audio))))
    (SUBJECT-0 (ID 2) (NP (ID 5) ( (NNP PTT) (NN switch))))
    (PP (ID 6) (PREP (IN on)) (NP (ID 10) ( (DT the) (NNP CIU))))
    (PARENT fails-12)))

(SI
  (fails-12
    (pred verb-ont fail-action)
    (theme (rel (id OBJECTS-0 17) (conj to) (clause transmit-17)))
    (inanimate-cause
      (np
        (np
          (id SUBJECT-0 5)
          (senses (thing (mod PTT) (head switch))))
        (pp
          (prep on)
          (np
            (id PP SUBJECT-0 0 10)
            (senses (thing (mod the) (head CIU))))))))
  (transmit-17
    (pred verb-ont transmit-transfer)
    (theme
      (np (id OBJECTS-0 19) (senses (thing (mod ) (head audio)))))
    (inanimate-cause
      (np
        (np
          (id SUBJECT-0 5)
          (senses (thing (mod PTT) (head switch))))
        (pp
          (prep on)
          (np
            (id PP SUBJECT-0 0 10)
            (senses (thing (mod the) (head CIU)))))))
    (pp-0
      (pp
        (prep due to)
        (np
          (np
            (id PP PP 0 26)
            (senses (thing (mod internal) (head failure))))
          (pp
            (prep of)
            (np
              (id PP PP 1 31)
              (senses (thing (mod the) (head switch)))))))))

SENT # 10

Left Control Valve fails to vent pneumatic cavity when commanded .

(S1 (S (NP (NNP Left) (NNP Control) (NNP Valve)) (VP (VBZ fails) (S-INF (VP (TO
to) (VP (VB vent) (NP (JJ pneumatic) (NN cavity)) (SBAR (WHADVP (WRB when)) (S
```

```
(VP (VBN commanded))))))))) (PERIOD .)))

(MCR
  (fails-7
    (VERB
      (MAIN-VERB fails fail)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0
      (ID 8)
      (RELATIVE (ID 12) (CONJ (TO to)) (CLAUSE vent-12)))
    (SUBJECT-0
      (ID 2)
      (NP (ID 5) ( (NNP Left) (NNP Control) (NNP Valve)))))
  (vent-12
    (VERB
      (MAIN-VERB vent vent)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 10) ( (TO to))))
      (TENSE VB))
    (OBJECTS-0 (ID 13) (NP (ID 15) ( (JJ pneumatic) (NN cavity))))
    (OBJECTS-1
      (ID 16)
      (RELATIVE (ID 21) (CONJ (WRB when)) (CLAUSE commanded-21)))
    (SUBJECT-0
      (ID 2)
      (NP (ID 5) ( (NNP Left) (NNP Control) (NNP Valve))))
    (PARENT fails-7))
  (commanded-21
    (VERB
      (MAIN-VERB commanded command)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBN))
    (SUBJECT-0 (ID 13) (NP (ID 15) ( (JJ pneumatic) (NN cavity))))
    (PARENT vent-12)))

(SI
  (fails-7
    (pred verb-ont fail-action)
    (theme (rel (id OBJECTS-0 12) (conj to) (clause vent-12)))
    (inanimate-cause
      (np
        (id SUBJECT-0 5)
        (senses (thing (mod Left Control) (head Valve))))))
  (vent-12
    (pred verb-ont nil)
    (obj-0
      (np
        (id OBJECTS-0 15)
        (senses (thing (mod pneumatic) (head cavity)))))
```

```
    (cp-1 (rel (id OBJECTS-1 21) (conj when) (clause commanded-21)))
    (subj
      (np
        (id SUBJECT-0 5)
        (senses (thing (mod Left Control) (head Valve))))))
  (commanded-21
    (pred verb-ont nil)
    (subj
      (np
        (id SUBJECT-0 15)
        (senses (thing (mod pneumatic) (head cavity)))))))


SENT # 24

The VDA shall generate an output drive signal that provides energy necessary to
fire the propulsion subsystem pyrotechnic isolation valve NSI .

(S1 (S (NP (DT The) (NNP VDA)) (VP (MD shall) (VP (VB generate) (NP-REL (NP (DT
an) (NN output) (NN drive) (NN signal)) (SBAR (WHNP (WDT that)) (S (VP (VBZ
provides) (S (NP (NN energy)) (ADJP (JJ necessary) (S (VP (TO to) (VP (VB fire)
(S (NP (DT the) (NN propulsion) (NN subsystem)) (NP (JJ pyrotechnic) (NN
isolation) (NN valve) (NNP NSI))))))))))))) (PERIOD .)))

(MCR
  (generate-8
    (VERB
      (MAIN-VERB generate generate)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 6) ( (MD shall))))
      (TENSE VB))
    (OBJECTS-0
      (ID 9)
      (NP (ID 14) ( (DT an) (NN output) (NN drive) (NN signal))))
    (RELATIVE (ID 20) (CONJ (WDT that)) (CLAUSE provides-20))
    (SUBJECT-0 (ID 2) (NP (ID 4) ( (DT The) (NNP VDA)))))
  (provides-20
    (VERB
      (MAIN-VERB provides provide)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (CLAUSE-TYPE RELATIVE)
      (TENSE VBZ))
    (OBJECTS-0 (ID 22) (NP (ID 23) ( (NN energy))))
    (OBJECTS-1 (ID 24) (ADJP (ID 25) ( (JJ necessary))))
    (RELATIVE (ID 30) (CONJ (TO to)) (CLAUSE fire-30))
    (SUBJECT-0
      (ID 10)
      (NP (ID 14) ( (DT an) (NN output) (NN drive) (NN signal))))
    (SUBJECT-1 (ID 2) (NP (ID 4) ( (DT The) (NNP VDA))))
    (PARENT generate-8))
  (fire-30
```

```
    (VERB
      (MAIN-VERB fire fire)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 28) ( (TO to))))
      (TENSE VB))
    (OBJECTS-0
      (ID 32)
      (NP (ID 35) ( (DT the) (NN propulsion) (NN subsystem))))
    (OBJECTS-1
      (ID 36)
      (NP
        (ID 40)
        ( (JJ pyrotechnic) (NN isolation) (NN valve) (NNP NSI))))
    (SUBJECT-0 (ID 22) (NP (ID 23) ( (NN energy))))
    (SUBJECT-1
      (ID 10)
      (NP (ID 14) ( (DT an) (NN output) (NN drive) (NN signal))))
    (SUBJECT-2 (ID 2) (NP (ID 4) ( (DT The) (NNP VDA))))
    (PARENT provides-20)))

(SI
  (generate-8
    (pred verb-ont nil)
    (obj-0
      (np
        (np
          (id OBJECTS-0 14)
          (senses (thing (mod an output drive) (head signal))))
        (rel
          (id RELATIVE OBJECTS-0 0 20)
          (conj that)
          (clause provides-20))))
    (subj
      (np (id SUBJECT-0 4) (senses (thing (mod The) (head VDA))))))
  (provides-20
    (pred verb-ont provide-transfer)
    (theme
      (np (id OBJECTS-0 23) (senses (thing (mod ) (head energy)))))
    (inanimate-cause
      (np
        (id SUBJECT-0 14)
        (senses (thing (mod an output drive) (head signal)))))
    (adjp-0 (adjp (id OBJECTS-1 25) (words necessary))))
  (fire-30
    (pred verb-ont fire-activate)
    (theme
      (np
        (id OBJECTS-0 35)
        (senses (thing (mod the propulsion) (head subsystem)))))
    (inanimate-cause
      (np (id SUBJECT-0 23) (senses (thing (mod ) (head energy)))))
    (obj-0
```

```
      (np
        (id OBJECTS-1 40)
        (senses (thing (mod pyrotechnic isolation valve) (head NSI)))))))

SENT # 25

The activation sequence sends a command to fire the pryotechnic device enabling
gas flow , fires thrusters to re-seat the thruster valves , and resets the rate
sensors .

(S1 (S (NP (DT The) (NN activation) (NN sequence)) (VP-COORD (VP (VBZ sends)
(NP (DT a) (NN command) (S (VP (TO to) (VP (VB fire) (NP-REL (NP (DT the) (JJ
pryotechnic) (NN device)) (VP (VBG enabling) (NP (NN gas) (NN flow)))))))))
(COMMA ,) (VP (VBZ fires) (S (NP (NNS thrusters)) (VP (TO to) (VP (VB re-seat)
(NP (DT the) (NN thruster) (NNS valves)))))) (COMMA ,) (CC and) (VP (VBZ
resets) (NP (DT the) (NN rate) (NNS sensors)))) (PERIOD .)))

(MCR
  (resets-44
    (VERB
      (MAIN-VERB resets reset)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0
      (ID 45)
      (NP (ID 48) ( (DT the) (NN rate) (NNS sensors))))
    (SUBJECT-0
      (ID 2)
      (NP (ID 5) ( (DT The) (NN activation) (NN sequence)))))
  (sends-8
    (VERB
      (MAIN-VERB sends send)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0 (ID 9) (NP (ID 11) ( (DT a) (NN command))))
    (RELATIVE (ID 16) (CONJ (TO to)) (CLAUSE fire-16))
    (SUBJECT-0
      (ID 2)
      (NP (ID 5) ( (DT The) (NN activation) (NN sequence)))))
  (fire-16
    (VERB
      (MAIN-VERB fire fire)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 14) ( (TO to))))
      (TENSE VB))
    (OBJECTS-0
      (ID 17)
      (NP (ID 21) ( (DT the) (JJ pryotechnic) (NN device))))
    (RELATIVE (ID 23) (CONJ ) (CLAUSE enabling-23))
    (SUBJECT-0
```

```
      (ID 2)
      (NP (ID 5) ( (DT The) (NN activation) (NN sequence))))
    (PARENT sends-8 fires-29 resets-44))
  (enabling-23
    (VERB
      (MAIN-VERB enabling enable)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (CLAUSE-TYPE REDUCED-RELATIVE)
      (TENSE VBG))
    (OBJECTS-0 (ID 24) (NP (ID 26) ( (NN gas) (NN flow))))
    (SUBJECT-0
      (ID 18)
      (NP (ID 21) ( (DT the) (JJ pryotechnic) (NN device))))
    (PARENT fire-16))
  (re-seat-36
    (VERB
      (MAIN-VERB re-seat re-seat)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 34) ( (TO to))))
      (TENSE VB))
    (OBJECTS-0
      (ID 37)
      (NP (ID 40) ( (DT the) (NN thruster) (NNS valves))))
    (SUBJECT-0 (ID 31) (NP (ID 32) ( (NNS thrusters))))
    (PARENT sends-8 fires-29 resets-44))
  (fires-29
    (VERB
      (MAIN-VERB fires fire)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0 (ID 30) (NP (ID 32) ( (NNS thrusters))))
    (RELATIVE (ID 36) (CONJ (TO to)) (CLAUSE re-seat-36))
    (SUBJECT-0
      (ID 2)
      (NP (ID 5) ( (DT The) (NN activation) (NN sequence))))))

(SI
  (resets-44
    (pred verb-ont nil)
    (obj-0
      (np
        (id OBJECTS-0 48)
        (senses (thing (mod the rate) (head sensors)))))
    (subj
      (np
        (id SUBJECT-0 5)
        (senses (thing (mod The activation) (head sequence))))))
  (sends-8
    (pred verb-ont nil)
    (obj-0
```

```
    (np
      (np
        (id OBJECTS-0 11)
        (senses (thing (mod a) (head command))))
      (rel (id RELATIVE OBJECTS-0 0 16) (conj to) (clause fire-16))))
  (subj
    (np
      (id SUBJECT-0 5)
      (senses (thing (mod The activation) (head sequence))))))
(fire-16
  (pred verb-ont fire-activate)
  (theme
    (np
      (np
        (id OBJECTS-0 21)
        (senses (thing (mod the pryotechnic) (head device))))
      (rel
        (id RELATIVE OBJECTS-0 0 23)
        (conj )
        (clause enabling-23))))
  (inanimate-cause
    (np
      (id SUBJECT-0 5)
      (senses (thing (mod The activation) (head sequence))))))
(enabling-23
  (pred verb-ont nil)
  (obj-0
    (np (id OBJECTS-0 26) (senses (thing (mod gas) (head flow)))))
  (subj
    (np
      (id SUBJECT-0 21)
      (senses (thing (mod the pryotechnic) (head device))))))
(re-seat-36
  (pred verb-ont nil)
  (obj-0
    (np
      (id OBJECTS-0 40)
      (senses (thing (mod the thruster) (head valves)))))
  (subj
    (np (id SUBJECT-0 32) (senses (thing (mod ) (head thrusters))))))
(fires-29
  (pred verb-ont fire-activate)
  (theme
    (np
      (np
        (id OBJECTS-0 32)
        (senses (thing (mod ) (head thrusters))))
      (rel
        (id RELATIVE OBJECTS-0 0 36)
        (conj to)
        (clause re-seat-36))))
  (inanimate-cause
    (np
```

```
        (id SUBJECT-0 5)
        (senses (thing (mod The activation) (head sequence)))))))

SENT # 27

The Abort Motor igniter provides initiation energy to the Abort Motor .

(S1 (S (NP (DT The) (NNP Abort) (NNP Motor) (NN igniter)) (VP (VBZ provides)
(NP (NN initiation) (NN energy)) (PP (TO to) (NP (DT the) (NN Abort) (NNP
Motor)))) (PERIOD .)))

(MCR
  (provides-8
    (VERB
      (MAIN-VERB provides provide)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (PP
      (ID 12)
      (PREP (TO to))
      (NP (ID 17) ( (DT the) (NN Abort) (NNP Motor))))
    (OBJECTS-0 (ID 9) (NP (ID 11) ( (NN initiation) (NN energy))))
    (SUBJECT-0
      (ID 2)
      (NP (ID 6) ( (DT The) (NNP Abort) (NNP Motor) (NN igniter))))))

(SI
  (provides-8
    (pred verb-ont provide-transfer)
    (theme
      (np
        (id OBJECTS-0 11)
        (senses (thing (mod initiation) (head energy)))))
    (goal
      (pp
        (prep to)
        (np
          (id PP PP 0 17)
          (senses (thing (mod the Abort) (head Motor))))))
    (inanimate-cause
      (np
        (id SUBJECT-0 6)
        (senses (thing (mod The Abort Motor) (head igniter)))))))

SENT # 85

Timeline Management monitors and controls the authorization for pyrotechnic
commands based on mission segments and phases .

(S1 (S (NP (NN Timeline) (NNP Management)) (VP-COORD (VBZ monitors) (CC and)
(VBZ controls) (NP (NP (DT the) (NN authorization)) (PP (IN for) (NP-REL (NP
(JJ pyrotechnic) (NNS commands)) (VP (VBN based) (PP (IN on) (NP-COORD (NN
```

```
mission) (NNS segments) (CC and) (NNS phases))))))))) (PERIOD .)))

(MCR
  (based-20
    (VERB
      (MAIN-VERB based base)
      (VERB-TYPE VERB)
      (VOICE PASSIVE)
      (CLAUSE-TYPE REDUCED-RELATIVE)
      (TENSE VBN))
    (PP
      (ID 21)
      (PREP (IN on))
      (NP
        (ID 23)
        (
          (AND
            (NP (ID 25) ( (NN mission) (NNS segments)))
            (NP (ID 27) ( (NNS phases)))))))
    (SUBJECT-0
      (ID 16)
      (NP (ID 18) ( (JJ pyrotechnic) (NNS commands))))
    (SUBJECT-1 (ID 10) (NP (ID 12) ( (DT the) (NN authorization))))
    (SUBJECT-2 (ID 2) (NP (ID 4) ( (NN Timeline) (NNP Management))))
    (PARENT monitors-6 controls-8))
  (monitors-6
    (VERB
      (MAIN-VERB monitors monitor)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0 (ID 9) (NP (ID 12) ( (DT the) (NN authorization))))
    (PP
      (ID 13)
      (PREP (IN for))
      (NP (ID 18) ( (JJ pyrotechnic) (NNS commands))))
    (RELATIVE (ID 20) (CONJ ) (CLAUSE based-20))
    (SUBJECT-0 (ID 2) (NP (ID 4) ( (NN Timeline) (NNP Management)))))
  (controls-8
    (VERB
      (MAIN-VERB controls control)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0 (ID 9) (NP (ID 12) ( (DT the) (NN authorization))))
    (PP
      (ID 13)
      (PREP (IN for))
      (NP (ID 18) ( (JJ pyrotechnic) (NNS commands))))
    (RELATIVE (ID 20) (CONJ ) (CLAUSE based-20))
    (SUBJECT-0 (ID 2) (NP (ID 4) ( (NN Timeline) (NNP Management))))))

(SI
```

```
(based-20
  (pred verb-ont nil)
  (obj-0
    (np
      (id SUBJECT-0 18)
      (senses (thing (mod pyrotechnic) (head commands)))))
  (pp-0
    (pp
      (prep on)
      (np
        (AND
          (np
            (id PP PP 0 25)
            (senses (thing (mod mission) (head segments))))
          (np
            (id PP PP 0 27)
            (senses (thing (mod ) (head phases)))))))))
(monitors-6
  (pred verb-ont nil)
  (obj-0
    (np
      (np
        (id OBJECTS-0 12)
        (senses (thing (mod the) (head authorization))))
      (pp
        (prep for)
        (np
          (np
            (id PP OBJECTS-0 0 18)
            (senses (thing (mod pyrotechnic) (head commands))))
          (rel (id RELATIVE PP-13 0 20) (conj ) (clause based-20))))))
  (subj
    (np
      (id SUBJECT-0 4)
      (senses (thing (mod Timeline) (head Management))))))
(controls-8
  (pred verb-ont control-manipulate)
  (theme
    (np
      (np
        (id OBJECTS-0 12)
        (senses (thing (mod the) (head authorization))))
      (pp
        (prep for)
        (np
          (np
            (id PP OBJECTS-0 0 18)
            (senses (thing (mod pyrotechnic) (head commands))))
          (rel (id RELATIVE PP-13 0 20) (conj ) (clause based-20))))))
  (inanimate-cause
    (np
      (id SUBJECT-0 4)
      (senses (thing (mod Timeline) (head Management)))))))
```

```
SENT # 89

Micro-RIU function is to collect and format vehicle structure sensor readings
into digital data for transfer to memory storage in the VMC .

(S1 (S (NP (NNP Micro-RIU) (NN function)) (VP (VBZ is) (S-INF (VP (TO to) (VP-COORD (VP (VB co
(MCR
  (collect-12
    (VERB
      (MAIN-VERB collect collect)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 9) ( (TO to))))
      (TENSE VB))
    (SUBJECT-0 (ID 2) (NP (ID 4) ( (NNP Micro-RIU) (NN function))))
    (PARENT is-6))
  (is-6
    (VERB
      (MAIN-VERB is be)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0
      (ID 7)
      (RELATIVE
        (AND
          (RELATIVE (ID 12) (CONJ (TO to)) (CLAUSE collect-12))
          (RELATIVE (ID 15) (CONJ (TO to)) (CLAUSE format-15)))))
    (SUBJECT-0 (ID 2) (NP (ID 4) ( (NNP Micro-RIU) (NN function)))))
  (format-15
    (VERB
      (MAIN-VERB format format)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 9) ( (TO to))))
      (TENSE VB-NN))
    (PP
      (ID 21)
      (PREP (IN into))
      (NP (ID 26) ( (JJ digital) (NNS data))))
    (PP (ID 27) (PREP (IN for)) (NP (ID 30) ( (NN transfer))))
    (PP
      (ID 31)
      (PREP (TO to))
      (NP (ID 36) ( (NN memory) (NN storage))))
    (PP (ID 37) (PREP (IN in)) (NP (ID 41) ( (DT the) (NNP VMC))))
    (OBJECTS-0
      (ID 16)
      (NP
        (ID 20)
        ( (NN vehicle) (NN structure) (NN sensor) (NNS readings))))
    (SUBJECT-0 (ID 2) (NP (ID 4) ( (NNP Micro-RIU) (NN function))))
```

```
    (PARENT is-6)))

(SI
  (collect-12
    (pred verb-ont collect)
    (inanimate-cause
      (np
        (id SUBJECT-0 4)
        (senses (thing (mod Micro-RIU) (head function))))))
  (is-6
    (pred verb-ont nil)
    (cp-0 (rel (id OBJECTS-0 12) (conj to) (clause collect-12)))
    (subj
      (np
        (id SUBJECT-0 4)
        (senses (thing (mod Micro-RIU) (head function))))))
  (format-15
    (pred verb-ont format-change)
    (theme
      (np
        (id OBJECTS-0 20)
        (senses
          (thing (mod vehicle structure sensor) (head readings)))))
    (to-state
      (pp
        (prep into)
        (np
          (np
            (id PP PP 0 26)
            (senses (thing (mod digital) (head data))))
          (pp
            (prep for)
            (np
              (id PP PP 1 30)
              (senses (thing (mod ) (head transfer))))))))
    (inanimate-cause
      (np
        (id SUBJECT-0 4)
        (senses (thing (mod Micro-RIU) (head function)))))
    (pp-2
      (pp
        (prep to)
        (np
          (np
            (id PP PP 2 36)
            (senses (thing (mod memory) (head storage))))
          (pp
            (prep in)
            (np
              (id PP PP 3 41)
              (senses (thing (mod the) (head VMC)))))))))

SENT # 93
```

```
This hardware includes filters , QDs , and manual valves used to fill and drain
the Dowforst , helium and ammonia in the CM , and to fill and drain the
Refrigerant HFE-7000 and R-134a loops in the SM .

(S1 (S (NP (DT This) (NN hardware)) (VP (VBZ includes) (NP-COORD (NP (NNS
filters)) (COMMA ,) (NP (NNP QDs)) (COMMA ,) (CC and) (NP-REL (NP (JJ manual)
(NNS valves)) (VP (VBN used) (S (VP-COORD (VP (TO to) (VP-COORD (VB fill) (CC
and) (VB drain) (NP-COORD (NP (DT the) (NNP Dowforst)) (COMMA ,) (NP (NN
helium)) (CC and) (NP (NN ammonia))) (PP (IN in) (NP (DT the) (NNP CM)))))
(COMMA ,) (CC and) (VP (TO to) (VP-COORD (VB fill) (CC and) (VB drain)
(NP-COORD (DT the) (NNP Refrigerant) (NNP HFE-7000) (CC and) (NNP R-134a) (NNS
loops)) (PP (IN in) (NP (DT the) (NN SM)))))))))) (PERIOD .)))

(MCR
  (drain-51
    (VERB
      (MAIN-VERB drain drain)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 47) ( (TO to))))
      (TENSE VB))
    (PP (ID 59) (PREP (IN in)) (NP (ID 63) ( (DT the) (NN SM))))
    (OBJECTS-0
      (ID 52)
      (NP
        (ID 52)
        (
          (AND
            (NP (ID 55) ( (DT the) (NNP Refrigerant) (NNP HFE-7000)))
            (NP (ID 58) ( (NNP R-134a) (NNS loops)))))))
    (SUBJECT-0 (ID 16) (NP (ID 18) ( (JJ manual) (NNS valves))))
    (SUBJECT-1 (ID 11) (NP (ID 12) ( (NNP QDs))))
    (SUBJECT-2 (ID 8) (NP (ID 9) ( (NNS filters))))
    (SUBJECT-3 (ID 2) (NP (ID 4) ( (DT This) (NN hardware))))
    (SUBJECT-4
      (ID 29)
      (NP
        (ID 29)
        (
          (AND
            (NP (ID 32) ( (DT the) (NNP Dowforst)))
            (NP (ID 35) ( (NN helium)))
            (NP (ID 38) ( (NN ammonia)))))))
    (PARENT used-20))
  (fill-26
    (VERB
      (MAIN-VERB fill fill)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 24) ( (TO to))))
      (TENSE VB))
    (PP (ID 39) (PREP (IN in)) (NP (ID 43) ( (DT the) (NNP CM))))
```

```
  (OBJECTS-0
    (ID 29)
    (NP
      (ID 29)
      (
        (AND
          (NP (ID 32) ( (DT the) (NNP Dowforst)))
          (NP (ID 35) ( (NN helium)))
          (NP (ID 38) ( (NN ammonia)))))))
  (SUBJECT-0 (ID 16) (NP (ID 18) ( (JJ manual) (NNS valves))))
  (SUBJECT-1 (ID 11) (NP (ID 12) ( (NNP QDs))))
  (SUBJECT-2 (ID 8) (NP (ID 9) ( (NNS filters))))
  (SUBJECT-3 (ID 2) (NP (ID 4) ( (DT This) (NN hardware))))
  (PARENT used-20))
(includes-6
  (VERB
    (MAIN-VERB includes include)
    (VERB-TYPE VERB)
    (VOICE ACTIVE)
    (TENSE VBZ))
  (OBJECTS-0
    (ID 7)
    (NP
      (ID 7)
      (
        (AND
          (NP (ID 9) ( (NNS filters)))
          (NP (ID 12) ( (NNP QDs)))
          (NP (ID 18) ( (JJ manual) (NNS valves)))))))
  (RELATIVE (ID 20) (CONJ ) (CLAUSE used-20))
  (SUBJECT-0 (ID 2) (NP (ID 4) ( (DT This) (NN hardware)))))
(used-20
  (VERB
    (MAIN-VERB used use)
    (VERB-TYPE VERB)
    (VOICE PASSIVE)
    (CLAUSE-TYPE REDUCED-RELATIVE)
    (TENSE VBN))
  (OBJECTS-0
    (ID 21)
    (RELATIVE
      (AND
        (RELATIVE (ID 26) (CONJ (TO to)) (CLAUSE fill-26))
        (RELATIVE (ID 28) (CONJ (TO to)) (CLAUSE drain-28))
        (RELATIVE (ID 49) (CONJ (TO to)) (CLAUSE fill-49))
        (RELATIVE (ID 51) (CONJ (TO to)) (CLAUSE drain-51)))))
  (SUBJECT-0 (ID 16) (NP (ID 18) ( (JJ manual) (NNS valves))))
  (SUBJECT-1 (ID 11) (NP (ID 12) ( (NNP QDs))))
  (SUBJECT-2 (ID 8) (NP (ID 9) ( (NNS filters))))
  (SUBJECT-3 (ID 2) (NP (ID 4) ( (DT This) (NN hardware))))
  (PARENT includes-6))
(drain-28
  (VERB
```

```
      (MAIN-VERB drain drain)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 24) ( (TO to))))
      (TENSE VB))
    (PP (ID 39) (PREP (IN in)) (NP (ID 43) ( (DT the) (NNP CM))))
    (OBJECTS-0
      (ID 29)
      (NP
        (ID 29)
        (
          (AND
            (NP (ID 32) ( (DT the) (NNP Dowforst)))
            (NP (ID 35) ( (NN helium)))
            (NP (ID 38) ( (NN ammonia)))))))
    (SUBJECT-0 (ID 16) (NP (ID 18) ( (JJ manual) (NNS valves))))
    (SUBJECT-1 (ID 11) (NP (ID 12) ( (NNP QDs))))
    (SUBJECT-2 (ID 8) (NP (ID 9) ( (NNS filters))))
    (SUBJECT-3 (ID 2) (NP (ID 4) ( (DT This) (NN hardware))))
    (PARENT used-20))
  (fill-49
    (VERB
      (MAIN-VERB fill fill)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 47) ( (TO to))))
      (TENSE VB))
    (PP (ID 59) (PREP (IN in)) (NP (ID 63) ( (DT the) (NN SM))))
    (OBJECTS-0
      (ID 52)
      (NP
        (ID 52)
        (
          (AND
            (NP (ID 55) ( (DT the) (NNP Refrigerant) (NNP HFE-7000)))
            (NP (ID 58) ( (NNP R-134a) (NNS loops)))))))
    (SUBJECT-0 (ID 16) (NP (ID 18) ( (JJ manual) (NNS valves))))
    (SUBJECT-1 (ID 11) (NP (ID 12) ( (NNP QDs))))
    (SUBJECT-2 (ID 8) (NP (ID 9) ( (NNS filters))))
    (SUBJECT-3 (ID 2) (NP (ID 4) ( (DT This) (NN hardware))))
    (SUBJECT-4
      (ID 29)
      (NP
        (ID 29)
        (
          (AND
            (NP (ID 32) ( (DT the) (NNP Dowforst)))
            (NP (ID 35) ( (NN helium)))
            (NP (ID 38) ( (NN ammonia)))))))
    (PARENT used-20)))

(SI
  (drain-51
```

```
  (pred verb-ont nil)
  (obj-0
    (np
      (AND
        (np
          (id OBJECTS-0 55)
          (senses (thing (mod the Refrigerant) (head HFE-7000))))
        (np
          (id OBJECTS-0 58)
          (senses (thing (mod R-134a) (head loops)))))))
  (subj
    (np
      (id SUBJECT-0 18)
      (senses (thing (mod manual) (head valves)))))
  (pp-0
    (pp
      (prep in)
      (np (id PP PP 0 63) (senses (thing (mod the) (head SM))))))
(fill-26
  (pred verb-ont nil)
  (obj-0
    (np
      (AND
        (np
          (id OBJECTS-0 32)
          (senses (thing (mod the) (head Dowforst))))
        (np
          (id OBJECTS-0 35)
          (senses (thing (mod ) (head helium))))
        (np
          (id OBJECTS-0 38)
          (senses (thing (mod ) (head ammonia)))))))
  (subj
    (np
      (id SUBJECT-0 18)
      (senses (thing (mod manual) (head valves)))))
  (pp-0
    (pp
      (prep in)
      (np (id PP PP 0 43) (senses (thing (mod the) (head CM))))))
(includes-6
  (pred verb-ont include)
  (thing-described
    (np
      (id SUBJECT-0 4)
      (senses (thing (mod This) (head hardware)))))
  (attribute-described
    (np
      (AND
        (np
          (id OBJECTS-0 9)
          (senses (thing (mod ) (head filters))))
        (np (id OBJECTS-0 12) (senses (thing (mod ) (head QDs))))
```

```
        (np
          (id OBJECTS-0 18)
          (senses (thing (mod manual) (head valves))))
        (rel (id RELATIVE OBJECTS-0 0 20) (conj ) (clause used-20))))))
(used-20
  (pred verb-ont nil)
  (cp-0 (rel (id OBJECTS-0 26) (conj to) (clause fill-26)))
  (obj-0
    (np
      (id SUBJECT-0 18)
      (senses (thing (mod manual) (head valves))))))
(drain-28
  (pred verb-ont nil)
  (obj-0
    (np
      (AND
        (np
          (id OBJECTS-0 32)
          (senses (thing (mod the) (head Dowforst))))
        (np
          (id OBJECTS-0 35)
          (senses (thing (mod ) (head helium))))
        (np
          (id OBJECTS-0 38)
          (senses (thing (mod ) (head ammonia)))))))
  (subj
    (np
      (id SUBJECT-0 18)
      (senses (thing (mod manual) (head valves)))))
  (pp-0
    (pp
      (prep in)
      (np (id PP PP 0 43) (senses (thing (mod the) (head CM)))))))
(fill-49
  (pred verb-ont nil)
  (obj-0
    (np
      (AND
        (np
          (id OBJECTS-0 55)
          (senses (thing (mod the Refrigerant) (head HFE-7000))))
        (np
          (id OBJECTS-0 58)
          (senses (thing (mod R-134a) (head loops)))))))
  (subj
    (np
      (id SUBJECT-0 18)
      (senses (thing (mod manual) (head valves)))))
  (pp-0
    (pp
      (prep in)
      (np (id PP PP 0 63) (senses (thing (mod the) (head SM))))))))
```

```
SENT # 96

This analysis includes evaluating the degree of isolation from 30/CD Hz to
400/CD Megahertz provided by the EPCE end item for power ripple and transients
to the equipment using isolated power .

(S1 (S (NP (DT This) (NN analysis)) (VP (VBZ includes) (S (VP (VBG evaluating) (NP (NP (DT the
(MCR
  (provided-31
    (VERB
      (MAIN-VERB provided provide)
      (VERB-TYPE VERB)
      (VOICE PASSIVE)
      (CLAUSE-TYPE REDUCED-RELATIVE)
      (TENSE VBN))
    (PP
      (ID 32)
      (PREP (IN by))
      (NP (ID 39) ( (DT the) (NNP EPCE) (NN end) (NN item))))
    (PP
      (ID 40)
      (PREP (IN for))
      (NP
        (ID 42)
        (
          (AND
            (NP (ID 44) ( (NN power) (NN ripple)))
            (NP (ID 46) ( (NNS transients)))))))
    (PP
      (ID 47)
      (PREP (TO to))
      (NP (ID 52) ( (DT the) (NN equipment))))
    (RELATIVE (ID 54) (CONJ ) (CLAUSE using-54))
    (POTENTIAL-SUBJECT-0
      (ID 34)
      (NP (ID 39) ( (DT the) (NNP EPCE) (NN end) (NN item))))
    (PP
      (ID 40)
      (PREP (IN for))
      (NP
        (ID 42)
        (
          (AND
            (NP (ID 44) ( (NN power) (NN ripple)))
            (NP (ID 46) ( (NNS transients)))))))
    (SUBJECT-0 (ID 27) (NP (ID 29) ( (CD 400/CD) (NNP Megahertz))))
    (SUBJECT-1 (ID 22) (NP (ID 23) ( (NNP Hz))))
    (SUBJECT-2 (ID 10) (NP (ID 13) ( (DT the) (NN degree))))
    (PP (ID 14) (PREP (IN of)) (NP (ID 17) ( (NN isolation))))
    (PARENT evaluating-9))
  (includes-6
    (VERB
      (MAIN-VERB includes include)
```

```
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0
      (ID 7)
      (RELATIVE (ID 9) (CONJ ) (CLAUSE evaluating-9)))
    (SUBJECT-0 (ID 2) (NP (ID 4) ( (DT This) (NN analysis)))))
  (using-54
    (VERB
      (MAIN-VERB using use)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (CLAUSE-TYPE REDUCED-RELATIVE)
      (TENSE VBG))
    (OBJECTS-0 (ID 55) (NP (ID 57) ( (JJ isolated) (NN power))))
    (SUBJECT-0 (ID 50) (NP (ID 52) ( (DT the) (NN equipment))))
    (SUBJECT-1 (ID 27) (NP (ID 29) ( (CD 400/CD) (NNP Megahertz))))
    (SUBJECT-2 (ID 22) (NP (ID 23) ( (NNP Hz))))
    (SUBJECT-3 (ID 10) (NP (ID 13) ( (DT the) (NN degree))))
    (PP (ID 14) (PREP (IN of)) (NP (ID 17) ( (NN isolation))))
    (PARENT provided-31))
  (evaluating-9
    (VERB
      (MAIN-VERB evaluating evaluate)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBG))
    (PP (ID 18) (PREP (IN from)) (NP (ID 21) ( (CD 30/CD))))
    (PP
      (ID 24)
      (PREP (TO to))
      (NP (ID 29) ( (CD 400/CD) (NNP Megahertz))))
    (RELATIVE (ID 31) (CONJ ) (CLAUSE provided-31))
    (OBJECTS-0 (ID 10) (NP (ID 13) ( (DT the) (NN degree))))
    (PP (ID 14) (PREP (IN of)) (NP (ID 17) ( (NN isolation))))
    (OBJECTS-1 (ID 22) (NP (ID 23) ( (NNP Hz))))
    (SUBJECT-0 (ID 2) (NP (ID 4) ( (DT This) (NN analysis))))
    (PARENT includes-6)))

(SI
  (provided-31
    (pred verb-ont provide-transfer)
    (theme
      (np
        (id SUBJECT-0 29)
        (senses (thing (mod 400/CD) (head Megahertz)))))
    (goal
      (pp
        (prep to)
        (np
          (np
            (id PP PP 2 52)
            (senses (thing (mod the) (head equipment))))
```

```
        (rel (id RELATIVE PP-47 0 54) (conj ) (clause using-54)))))
  (inanimate-cause
    (np
      (np
        (id POTENTIAL-SUBJECT-0 39)
        (senses (thing (mod the EPCE end) (head item))))
      (pp
        (prep for)
        (np
          (np
            (id PP POTENTIAL-SUBJECT-0 0 44)
            (senses (thing (mod power) (head ripple))))
          (pp
            (prep for)
            (np
              (id PP POTENTIAL-SUBJECT-0 0 46)
              (senses (thing (mod ) (head transients)))))))))
(includes-6
  (pred verb-ont include)
  (thing-described
    (np
      (id SUBJECT-0 4)
      (senses (thing (mod This) (head analysis)))))
  (attribute-described
    (rel (id OBJECTS-0 9) (conj ) (clause evaluating-9))))
(evaluating-9
  (pred verb-ont nil)
  (obj-0
    (np
      (np
        (id OBJECTS-0 13)
        (senses (thing (mod the) (head degree))))
      (pp
        (prep of)
        (np
          (id PP OBJECTS-0 0 17)
          (senses (thing (mod ) (head isolation)))))))
  (obj-1 (np (id OBJECTS-1 23) (senses (thing (mod ) (head Hz)))))
  (subj
    (np
      (id SUBJECT-0 4)
      (senses (thing (mod This) (head analysis)))))
  (pp-0
    (pp
      (prep from)
      (np (id PP PP 0 21) (senses (thing (mod ) (head 30/CD))))))
  (pp-1
    (pp
      (prep to)
      (np
        (np
          (id PP PP 1 29)
          (senses (thing (mod 400/CD) (head Megahertz))))
```

```
          (rel (id RELATIVE PP-24 0 31) (conj ) (clause provided-31))))))
  (using-54
    (pred verb-ont nil)
    (obj-0
      (np
        (id OBJECTS-0 57)
        (senses (thing (mod isolated) (head power)))))
    (subj
      (np
        (id SUBJECT-0 52)
        (senses (thing (mod the) (head equipment)))))))

SENT # 101

The SCCA 's main controller card hosts higher layer functions .

(S1 (S (NP (NP (DT The) (NNP SCCA) (POS 's)) (JJ main) (NN controller) (NN
card)) (VP (VBZ hosts) (NP (JJR higher) (NN layer) (NNS functions))) (PERIOD
.)))

(MCR
  (hosts-11
    (VERB
      (MAIN-VERB hosts host)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0
      (ID 12)
      (NP (ID 15) ( (JJR higher) (NN layer) (NNS functions))))
    (SUBJECT-0
      (ID 2)
      (NP
        (ID 9)
        ( (DT The) (NNP SCCA) (JJ main) (NN controller) (NN card))))))

(SI
  (hosts-11
    (pred verb-ont host-contain)
    (theme
      (np
        (id OBJECTS-0 15)
        (senses (thing (mod higher layer) (head functions)))))
    (at-loc
      (np
        (id SUBJECT-0 9)
        (senses (thing (mod The SCCA main controller) (head card)))))))

SENT # 102

All flight sw applications are hosted on the shared resources of the VMC .

(S1 (S (NP (DT All) (NN flight) (NN sw) (NNS applications)) (VP (AUX are) (VP
```

```
(VBN hosted) (PP (IN on) (NP (NP (DT the) (VBN shared) (NNS resources)) (PP (IN
of) (NP (DT the) (NNP VMC))))))) (PERIOD .)))

(MCR
  (hosted-10
    (VERB
      (MAIN-VERB hosted host)
      (VERB-TYPE VERB)
      (VOICE PASSIVE)
      (MODIFIERS ( (ID 8) ( (AUX are))))
      (TENSE VBN))
    (PP
      (ID 11)
      (PREP (IN on))
      (NP (ID 17) ( (DT the) (VBN shared) (NNS resources))))
    (PP (ID 18) (PREP (IN of)) (NP (ID 22) ( (DT the) (NNP VMC))))
    (SUBJECT-0
      (ID 2)
      (NP (ID 6) ( (DT All) (NN flight) (NN sw) (NNS applications))))))

(SI
  (hosted-10
    (pred verb-ont host-contain)
    (theme
      (np
        (id SUBJECT-0 6)
        (senses (thing (mod All flight sw) (head applications)))))
    (at-loc
      (pp
        (prep on)
        (np
          (np
            (id PP PP 0 17)
            (senses (thing (mod the shared) (head resources))))
          (pp
            (prep of)
            (np
              (id PP PP 1 22)
              (senses (thing (mod the) (head VMC)))))))))

SENT # 107

Prefer to replace unit to maintain quick crew notification of urgent situations
.

(S1 (S (VP (VBP Prefer) (S-INF (VP (TO to) (VP (VB replace) (NP (NN unit))
(S-INF (VP (TO to) (VP (VB maintain) (NP (NP (JJ quick) (NN crew) (NN
notification)) (PP (IN of) (NP (JJ urgent) (NNS situations))))))))))) (PERIOD
.)))

(MCR
  (maintain-15
    (VERB
```

```
      (MAIN-VERB maintain maintain)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 13) ( (TO to))))
      (TENSE VB))
    (OBJECTS-0
      (ID 16)
      (NP (ID 20) ( (JJ quick) (NN crew) (NN notification))))
    (PP
      (ID 21)
      (PREP (IN of))
      (NP (ID 25) ( (JJ urgent) (NNS situations))))
    (SUBJECT-0 (ID 9) (NP (ID 10) ( (NN unit))))
    (PARENT replace-8))
  (Prefer-3
    (VERB
      (MAIN-VERB Prefer prefer)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBP))
    (OBJECTS-0
      (ID 4)
      (RELATIVE (ID 8) (CONJ (TO to)) (CLAUSE replace-8))))
  (replace-8
    (VERB
      (MAIN-VERB replace replace)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 6) ( (TO to))))
      (TENSE VB))
    (OBJECTS-0 (ID 9) (NP (ID 10) ( (NN unit))))
    (OBJECTS-1
      (ID 11)
      (RELATIVE (ID 15) (CONJ (TO to)) (CLAUSE maintain-15)))
    (PARENT Prefer-3)))

(SI
  (maintain-15
    (pred verb-ont maintain)
    (theme
      (np
        (np
          (id OBJECTS-0 20)
          (senses (thing (mod quick crew) (head notification))))
        (pp
          (prep of)
          (np
            (id PP OBJECTS-0 0 25)
            (senses (thing (mod urgent) (head situations)))))))
    (subj (np (id SUBJECT-0 10) (senses (thing (mod ) (head unit))))))
  (Prefer-3
    (pred verb-ont nil)
    (cp-0 (rel (id OBJECTS-0 8) (conj to) (clause replace-8))))
```

```
  (replace-8
    (pred verb-ont nil)
    (obj-0
      (np (id OBJECTS-0 10) (senses (thing (mod ) (head unit)))))
    (cp-1 (rel (id OBJECTS-1 15) (conj to) (clause maintain-15)))))


SENT # 110

Current design incorporates desiccants to maintain the internal humidity to
insure proper JM performance .

(S1 (S (NP (JJ Current) (NN design)) (VP (VBZ incorporates) (S (NP (NNS
desiccants)) (VP (TO to) (VP (VB maintain) (NP (DT the) (JJ internal) (NN
humidity)) (S-INF (VP (TO to) (VP (VB insure) (NP (JJ proper) (NNP JM) (NN
performance)))))))) (PERIOD .)))

(MCR
  (insure-22
    (VERB
      (MAIN-VERB insure insure)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 20) ( (TO to))))
      (TENSE VB))
    (OBJECTS-0
      (ID 23)
      (NP (ID 26) ( (JJ proper) (NNP JM) (NN performance))))
    (SUBJECT-0 (ID 8) (NP (ID 9) ( (NNS desiccants))))
    (SUBJECT-1
      (ID 14)
      (NP (ID 17) ( (DT the) (JJ internal) (NN humidity))))
    (PARENT maintain-13))
  (incorporates-6
    (VERB
      (MAIN-VERB incorporates incorporate)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0 (ID 7) (NP (ID 9) ( (NNS desiccants))))
    (RELATIVE (ID 13) (CONJ (TO to)) (CLAUSE maintain-13))
    (SUBJECT-0 (ID 2) (NP (ID 4) ( (JJ Current) (NN design)))))
  (maintain-13
    (VERB
      (MAIN-VERB maintain maintain)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 11) ( (TO to))))
      (TENSE VB))
    (OBJECTS-0
      (ID 14)
      (NP (ID 17) ( (DT the) (JJ internal) (NN humidity))))
    (OBJECTS-1
```

```
      (ID 18)
      (RELATIVE (ID 22) (CONJ (TO to)) (CLAUSE insure-22)))
    (SUBJECT-0 (ID 8) (NP (ID 9) ( (NNS desiccants))))
    (PARENT incorporates-6)))

(SI
  (incorporates-6
    (pred verb-ont nil)
    (obj-0
      (np
        (np
          (id OBJECTS-0 9)
          (senses (thing (mod ) (head desiccants))))
        (rel
          (id RELATIVE OBJECTS-0 0 13)
          (conj to)
          (clause maintain-13))))
    (subj
      (np
        (id SUBJECT-0 4)
        (senses (thing (mod Current) (head design))))))
  (insure-22
    (pred verb-ont nil)
    (obj-0
      (np
        (id OBJECTS-0 26)
        (senses (thing (mod proper JM) (head performance)))))
    (subj
      (np (id SUBJECT-0 9) (senses (thing (mod ) (head desiccants))))))
  (maintain-13
    (pred verb-ont maintain)
    (theme
      (np
        (id OBJECTS-0 17)
        (senses (thing (mod the internal) (head humidity)))))
    (cp-0 (rel (id OBJECTS-1 22) (conj to) (clause insure-22)))
    (subj
      (np (id SUBJECT-0 9) (senses (thing (mod ) (head desiccants)))))))

SENT # 115

The EDE is currently maintaining channel margin for sensors and valve
interfaces since the EDE card is a common card in the six PDU locations .

(S1 (S (NP (DT The) (NN EDE)) (VP (AUX is) (ADVP (RB currently)) (VP (VBG
maintaining) (NP (NP (NN channel) (NN margin)) (PP (IN for) (NP-COORD (NNS
sensors) (CC and) (NN valve) (NNS interfaces)))) (SBAR (IN since) (S (NP (DT
the) (JJ EDE) (NN card)) (VP (VBZ is) (NP (NP (DT a) (JJ common) (NN card)) (PP
(IN in) (NP (DT the) (CD six) (NNP PDU) (NNS locations))))))))) (PERIOD .)))

(MCR
  (is-30
    (VERB
```

```
      (MAIN-VERB is be)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (TENSE VBZ))
    (OBJECTS-0 (ID 31) (NP (ID 35) ( (DT a) (JJ common) (NN card))))
    (PP
      (ID 36)
      (PREP (IN in))
      (NP (ID 42) ( (DT the) (CD six) (NNP PDU) (NNS locations))))
    (SUBJECT-0 (ID 25) (NP (ID 28) ( (DT the) (JJ EDE) (NN card))))
    (PARENT maintaining-10))
  (maintaining-10
    (VERB
      (MAIN-VERB maintaining maintain)
      (VERB-TYPE VERB)
      (VOICE ACTIVE)
      (MODIFIERS ( (ID 7) ( (RB currently))) ( (ID 6) ( (AUX is))))
      (TENSE VBG))
    (OBJECTS-0 (ID 11) (NP (ID 14) ( (NN channel) (NN margin))))
    (PP
      (ID 15)
      (PREP (IN for))
      (NP
        (ID 17)
        (
          (AND
            (NP (ID 18) ( (NNS sensors)))
            (NP (ID 21) ( (NN valve) (NNS interfaces)))))))
    (OBJECTS-1
      (ID 22)
      (RELATIVE (ID 30) (CONJ (IN since)) (CLAUSE is-30)))
    (SUBJECT-0 (ID 2) (NP (ID 4) ( (DT The) (NN EDE))))))

(SI
  (is-30
    (pred verb-ont nil)
    (obj-0
      (np
        (np
          (id OBJECTS-0 35)
          (senses (thing (mod a common) (head card))))
        (pp
          (prep in)
          (np
            (id PP OBJECTS-0 0 42)
            (senses (thing (mod the six PDU) (head locations)))))))
    (subj
      (np
        (id SUBJECT-0 28)
        (senses (thing (mod the EDE) (head card))))))
  (maintaining-10
    (pred verb-ont maintain)
    (theme
```

```
  (np
    (np
      (id OBJECTS-0 14)
      (senses (thing (mod channel) (head margin))))
    (pp
      (prep for)
      (np
        (np
          (id PP OBJECTS-0 0 18)
          (senses (thing (mod ) (head sensors))))
        (pp
          (prep for)
          (np
            (id PP OBJECTS-0 0 21)
            (senses (thing (mod valve) (head interfaces)))))))))
(cp-0 (rel (id OBJECTS-1 30) (conj since) (clause is-30)))
(subj
  (np (id SUBJECT-0 4) (senses (thing (mod The) (head EDE)))))))
```

Appendix B. STAT Tutorial

STAT Tutorial

Version 1.0
7/22/2011

This document takes you through the steps for processing a new data set in STAT/Flamenco+. It will show you how to write a specification file for your dataset, how to run the data, how to view the STAT output, and how to view the results in Flamenco+.

Forward: We assume you have access to a system on which STAT and Flamenco+ have been installed, and you have the appropriate file permissions.

A note on file locations – The STAT User's Guide discusses how to install STAT in a standard configuration under Centos 5.4. In that configuration, all files are read from and written to the local machine. Section 2.2.2 *The whereami file,* discusses how to customize an installation if, for example, the html outputs are written to a network location for display by a dedicated server. This tutorial assumes the standard configuration.

The scripts which install the Centos 5.4 distribution of STAT put its Perl source-code files into /usr/lib/perl5/5.8.8/STAT. (As the "analyst" user, entering "Stat" at the command line will take you to that directory.) If you are using that configuration, the following does not apply to you.

However, users may wish to move some of the installed files. For instance, the installation scripts download and install a copy of the Apache web server onto the localhost machine. Instead of using that local version, you may wish to use an institutional webserver, which serves pages from a network drive, allowing anyone in your org to view them. Similarly, you may have unpacked the STAT source files onto a network drive. If you do that, you need to tell STAT the root of the directory where to write its output pages (something like /networkserver/httpd/html/projects) and the URL that location is served as (http://OurWebServer.example.com/projects), and so on.

Go to the directory where you unpacked the STAT source-code; in the Centos 5.4 distribution that would be /usr/local/perl5/5.8.8/STAT. From there, open the file base/whereami.pm. Follow the directions therein to edit the file in order to point STAT to the right places.

Writing the Spec File

Suppose that you have several related datasets called *shortProblems,* which have the same format but different data.

You need to create a spec file, *shortProblems.pm,* which will specify:

- Where to find the input file
- Which input fields hold the Record-ID and the Record-date
- Which fields to tag, and in which ontologies
- What fields to graph in the Overview page
- What field to graph and how to display an individual record in the LeafNode pages

- How the Flamenco+ pages should appear
- Where to write the output

The STAT distribution includes a practice file, shortProblems01.tsv. In the standard configuration, this is in `/var/www/html/ontologies/sources/test/datedExamples/`. This file contains 66 records, with artificial text and data. Notice the headers of this file:

- ID Number – An identifying number, each one unique
- Injury – Taken from a small set of values, or blank
- DOW – Day of Week
- Tcode – One of a small set of values
- Crit – Criticality of 1,2,3, or blank
- Initiation Date – The initiation date, dd/mm/yyyy
- Closed – The closure date, dd/mm/yyyy or blank
- Longitude and Latitude – The location of the incident
- Description – A sentence describing the problem; possibly more than one sentence or a non-sentence, like a noun-phrase
- PartName – The name of a piece of equipment associated with this report

You will write the spec to do the following:

- Tag the Description field for Failures
- Tag the PartName field for Equipment
- Produce bar-charts where the X-Axis is Time, as recorded in the Initiation Date
- View Injury, DOW, Tcode and Crit as facets in the Flamenco+ browsing.
- In the Flamenco+ item-view, see the tagged Description and PartName, the Longitude and Latitude, and the Initiation and Closed dates

The spec file is itself a Perl file. STAT contains a data structure, `%runSwitches`, which holds the specifications. Your spec file sets the values of `%runSwitches`.

The Finished Spec

Here's what the finished spec will look like. We'll go through it, line by line.

```
#!/usr/local/bin/perl

# A cutdown version of shortProblem.pm, trying to get a minimal set of specs.

use strict;
use vars qw(%runSwitches);

$runSwitches{proj}{sh1} =
    # Basic info
   {ID_Field    => 'ID Number', # The key field for the data
    dateField   => 'Initiation Date', # Field holding the date for trending.
    closeDateField              # Having dateField and closeDateField
      => 'Closed',              #   => Aging chart in overview.
    printLeafnodes      => 1,   # Causes the LeafNode pages to be printed
    printCharts         => 1,   # In LeafNode, causes graphs to be printed.
    reportsAre          => 'Incident Report',

     # Spec the tagging
    tagFieldHash  # Use these ontologies to tag these fields.
       => {FAILURE => {Description => 1},
           NOUN => {PartName => 1}},
      # For these ontologies, don't start with the root node.
      # Instead, only tag below these daughter nodes.
      # For these ontologies, don't start with the root node.
      # Instead, only tag below these daughter nodes.
    startWithNode =>
            {NOUN    => [qw(Equipment_or_Implement_or Equipment_Part
                        Physical_Interface_Compon Functional_Substance)],
             FAILURE => [qw(Impaired_Controllability Incompatible Ineffective
                        Mechanically_Impaired Input_Output_Deviation
                        Agent_Deviation_or_Error Functional_Deviation_or_E
                        Process_Deviation_or_Erro Resource_Use_Deviation
                        Artifact_Problem Not_Robust
                        Damaged_or_Injured_or_Des Damage_or_Impairment_Sour
                        Object_Conformity_Problem)]},

    # Spec the Overview page.
    overviewArgs        => {attrsForTop_N_Charts => [qw(Tcode DOW)],
                            discrepancyPlots => ['Injury']},

    # These apply both to Overview graphs and LeafNode graphs:
    fetch_stripeFn =>       # How to get the values for the attrByQtr
       \&stripedParamValFromSquareInputs,
    fetch_xFn      => \&yrQtrStringed, # How to format the date into quarters.
    x_spanParamPhrase =>  'Qtr',  # What to call the time units

    # Spec the Atlas pages.
    AtlasFields         => ['Description'],
```

```
# Spec the LeafNode pages
frontFunctions =>          # Invoke the function which prints the charts.
   [\&TrendLeafnodeChartsCGI],
  # Within the LeafNode page, spec what an individual record looks like
 nodeArgs =>
 {singleRecordPrintFn => \&H_catMapWdIn,  # Use this fn to print out one
 record
  long_fields              # Long_fields get a full line to themselves.
     => ['Description', 'PartName'],
  short_fields  => ['Initiation Date',  # Print out up to 4 in a line.
                    qw(Tcode Latitude Longitude Injury Crit Closed)],
  topcolumns     # The fields which get graphs in &TrendLeafnodeCharts
       => [qw(DOW)]},

# Spec the Flamenco pages
 flamencoArgs =>
  {dumpFlamenco    => 1,  # Create the Flamenco+ files.
   includeRoots => {NOUN => 1,
                    FAILURE => 1},
    attrFields  =>      # Fields to get included in the items file, but
       [qw(Latitude Longitude Closed)],    #  not (necessarily) facets
    facetsNotTagged =>   # Fields which become 1-level-deep facets.
       [qw(Tcode Injury DOW Crit)]}};


1  # Every Perl file ends with 1
```

Writing the Spec File

In a text editor, create the shortProblems.pm file in the Trend/proj directory. (In the standard configuration, this will be /usr/lib/perl5/5.8.8/STAT/Trend/proj, and there's already a copy there). Begin it a

```
#!/usr/bin/perl
use strict;
use vars qw(%runSwitches);
$runSwitches{project}{shortProblems} =
  {
```

Begin entering key/value pairs. Start with the spec for the ID_Field, dateField and closeDateField.

```
   ID_Field     => 'ID Number', # The key field for the data
   dateField         => 'Initiation Date',
   # Having dateField and closeDateField => Aging chart in Overview.
   closeDateField => 'Closed',
   printLeafnodes => 1, # Causes the LeafNode pages to be printed
   printCharts    => 1, # Print graphs in LeafNode pages
   reportsAre     => 'Incident Report', # Graphs label records as this
```

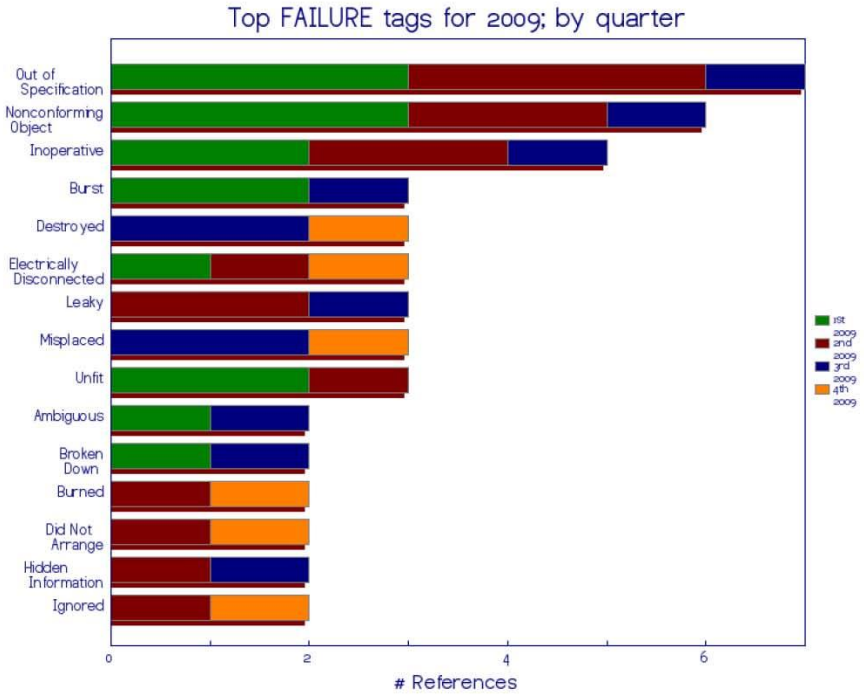
Specify which fields get tagged with which ontology.

```
   tagFieldHash  => {FAILURE => {Description => 1},
                     NOUN => {PartName => 1}},
```

The next spec is a little tricky to understand. The tops of the ontologies are very general – with categories like 'thing' for the Noun ontology and 'Problem' for the Failure ontology. Tags at that level are not very useful, and make it difficult to navigate through the ontologies. So instead, we only tag things which match concepts within the more-specific parts of the ontology. STAT will only tag for these concepts, and for their descendent-concepts (i.e., more specific concepts:

```
# For these ontologies, don't start with the root node.
# Instead, only tag below these daughter nodes.
   startWithNode =>
      {NOUN    => [qw(Equipment_or_Implement_or Equipment_Part
                  Physical_Interface_Compon Functional_Substance)],
       FAILURE => [qw(Impaired_Controllability Incompatible Ineffective
                  Mechanically_Impaired Input_Output_Deviation
                  Agent_Deviation_or_Error Functional_Deviation_or_E
                  Process_Deviation_or_Erro Resource_Use_Deviation
                  Artifact_Problem Not_Robust
                  Damaged_or_Injured_or_Des Damage_or_Impairment_Sour
                  Object_Conformity_Problem)]},
```

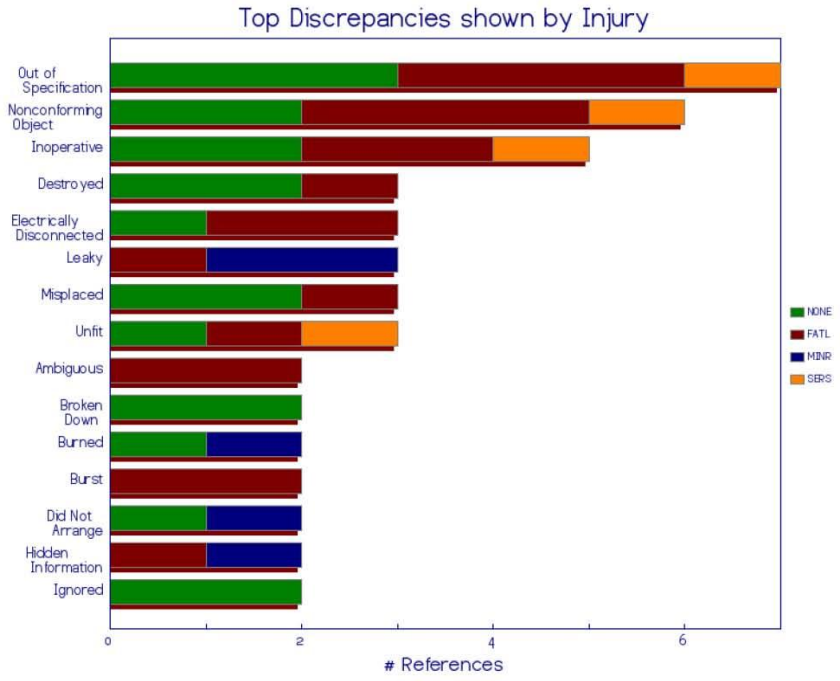
There will be an Overview page. By default, it will include charts for the most abundant concept-tags in each ontology, striped by the calendar-quarter of the associated records. For example:



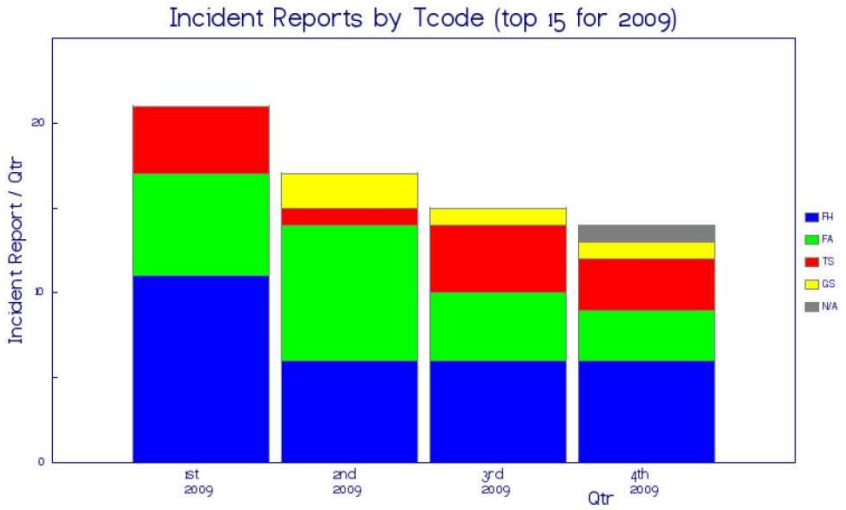
You may spec that these striped by any of the other fields of the input data. Putting this in the spec:

```
overviewArgs => {discrepancyPlots => ['Injury'],
                 attrsForTop_N_Charts => [qw(Tcode DOW)]},
```

Generates this chart for the *discrepancyPlots*:



That spec also generates this chart for Tcode (the similar chart for DOW is generated, but not shown here):



There are graphs generated both on the Overview page and on the LeafNode pages. Building the charts requires a (customizable) Perl function which fetches the data. The date-data needs to be converted from mm/dd/yyyy format into a Qtr/Yr format. And the charting function must be told what time-label to use.

```
# These apply both to Overview graphs and LeafNode graphs:
fetch_stripeFn # How to get the values for the attrByQtr
   => \&stripedParamValFromSquareInputs,
fetch_xFn     => \&yrQtrStringed, # How to format the date into quarters.
x_spanParamPhrase =>  'Qtr',
```

The Atlas pages list all the records and how they were tagged, with hyperlinks to very detailed pages for debugging. Spec which fields will be listed:

```
AtlasFields              => ['Description'],
```

And produce a Atlas page like:

ID Number	Parse Internals	UCF Parse	Description
N-1	Parse Internals	UCF Parse	He the inserts are broke down ; . requested by Mark Flood .
N-2	Parse Internals	UCF Parse	The unclean hard drive , S\/N 12312, in the control rm was detonated by absent information . If the delay line is severed , then the oscilloscope is not grounded .

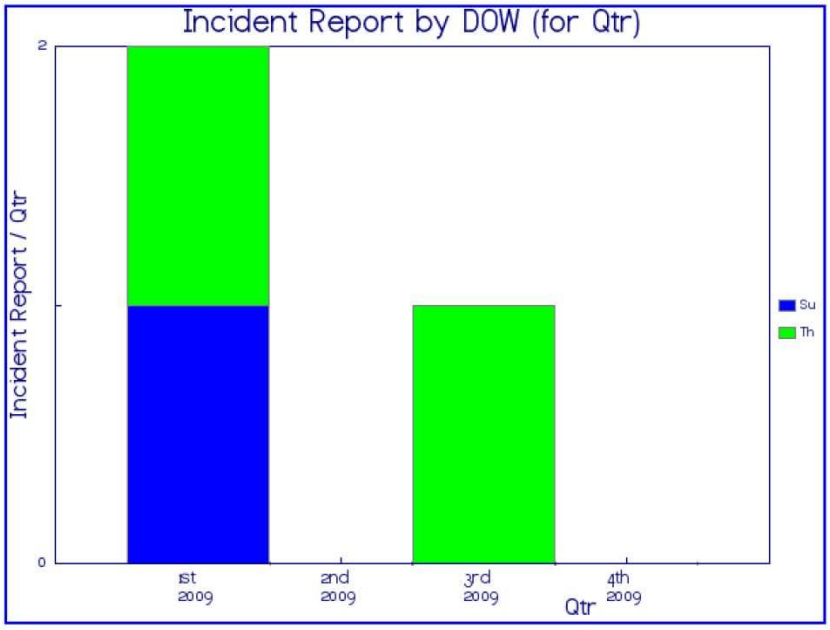
A LeafNode page is produced for each concept in an ontology, if any record was tagged with that concept. Specify how these pages appear. First spec a function which will print some charts at the top of the page:

```
frontFunctions => [\&TrendLeafnodeChartsCGI],
```

Within the LeafNode page, spec what an individual record looks like:

```
nodeArgs =>
{singleRecordPrintFn => \&H_catMapWdIn,# Use this fn to print out one record
 long_fields              # Long_fields get a full line to themselves.
    => ['Description', 'PartName'],
 short_fields  => ['Initiation Date',  # Print out up to 4 in a line.
                   qw(Tcode Latitude Longitude Injury Crit Closed)],
 topcolumns     # The fields which get graphs in &TrendLeafnodeCharts
      => [qw(DOW)]},
```

For the LeafNode page for *Burst*, this produces the header-chart:



And a table of individual records; here's one:

2	**ID Number**: N-2	**Initiation Date**: 1/5/2009	**Tcode**: FH	**Latitude**: 353201N
	Longitude: C973829W	**Injury**: FATL	**Crit**: 2	**Closed**: 1/20/2009
	Description: The unclean hard drive , SWN 12312, in the control rm was detonated by absent information . If the delay line is severed , then the oscilloscope is not grounded .			
	PartName: LOWER USS-02 ASSEMBLY			

The hyperlinked '2' (far left column of the table above) points to the Atlas page for the record N-2.

9

Spec the flamenco output files:

```
flamencoArgs =>
  {dumpFlamenco    => 1,  # Create the Flamenco+ files.
   includeRoots => {NOUN => 1,
                    FAILURE => 1},
    attrFields  =>       # Fields to get included in the items file, but
       [qw(Latitude Longitude Closed)],    #  not (necessarily) facets
    facetsNotTagged =>   # Fields which become 1-level-deep facets.
       [qw(Tcode Injury DOW Crit)]}};
```

This spec produces a Flamenco+ page with six facets: the tagged PartName and Description fields are each one facet, and the fields *Tcode Injury DOW Crit* were spec'd by *facetsNotTagged*.

Refine your search within these categories:

PARTNAME (group results)

Equipment_or_Implement_or (44)	NoRelevantTag (7)
Equipment_Part (15)	UnTagged (7)
Functional_Substance (8)	Physical_Interface_Compon (4)

DESCRIPTION (group results)

Damaged_or_Injured_or_Des (20)	Ineffective (7)
Input_Output_Deviation (13)	Artifact_Problem (7)
Functional_Deviation_or_E (13)	Process_Deviation_or_Erro (5)
Damage_or_Impairment_Sour (12)	Mechanically_Impaired (4)
UnTagged (11)	more...
Object_Conformity_Problem (10)	

TCODE

FH (28)	GS (4)
FA (21)	N/A (1)
TS (12)	

INJURY (group results)

NONE (26)	SERS (7)
FATL (25)	MINR (6)

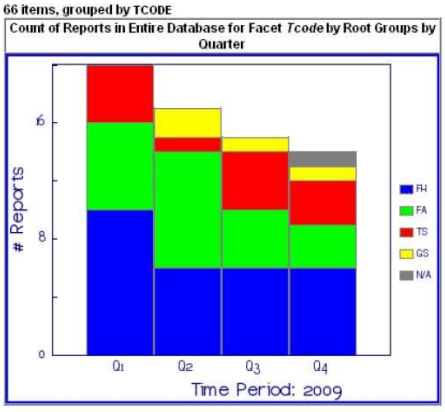
DOW (group results)

Sa (15)	Fr (9)
Tu (11)	Su (7)
We (9)	Mo (6)
Th (9)	

CRIT (group results)

3 (32)	1 (9)
2 (15)	

66 items, grouped by TCODE



In Flamenco+, users may navigate to an individual record. The example spec will produce an individual record with seven fields. The ID and Date are always shown first, and the tagged fields are shown last. The *attrFields* spec'd *Latitude Longitude Closed*. So an individual record appears as:

IDNumber: 16
Date: 4/23/2009
Latitude: 454259N
Longitude: 1205904W
Closed:
FAILURE_Description_text: The filter leaking had flawed scorching .
NOUN_PartName_text: FOAM

Running the Spec

Logged in as *analyst*, in a terminal window, go to the directory where the STAT source files are. In the standard configuration, this is /usr/lib/perl5/5.8.8/STAT. Your ~/.tcshrc file alias the command *Stat* to cd you to there:

```
% Stat; pwd
/usr/lib/perl5/5.8.8/STAT
```

Run the *trend* script, using your spec file on the example data:

```
% perl -w Trend/trend.pl -proj shortProblems -case shortProblems01
```

Output to Terminal during run

You should see output as:

```
Loading the case  spec from Trend/proj/shortProblems.pm
Creating directory
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X
Creating directory
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X/sc
ratch
   *main::RPT Reporting3 to
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X/sc
ratch/warns.html
   *main::INTEG Reporting3 to
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X/sc
ratch/UnknownsReport.html
   SCRATCH Reporting to
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X/sc
ratch/nattering.txt
Created directories beneath
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X :
atlas directory graphs flamenco leafnode parsInt
Ontology
Ontology from ontoPl dump
/user6/httpd/htdocs/projects/reconciler/plDumpOntologies/Vers 1.04
Aerospace Ontology.pl
     and
/user6/httpd/htdocs/projects/reconciler/plDumpOntologies/1.04_wordCache.pl
% Reloading ontStruct
```

```
   Opening
/user6/httpd/htdocs/projects/ontologies/sources/test/datedExamples/shortPr
oblems01.tsv as an input.
Creating directory
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X/..
/ucfParse
Not loading a Treebank cache; didn't find
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/ucfP
arse/IndexTreebank_DumpFile.pl
  Starting UCF Parser
Calling parser on text for field Description...
Calling parser on text for field PartName...
Indexing and evoking on onto: NOUN - field: PartName.
Indexing and evoking on onto: FAILURE - field: Description
 Done ucfParsing.
Dumping treebank to
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/ucfP
arse/IndexTreebank_DumpFile.pl ..
Wrote UCF parses to
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X/uc
fParse/ucf_parse.html
Creating directory
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X/uc
fParse
REporting to <a
href="http://tommy.jsc.nasa.gov/projects/reconciler/shortProblems/shortPro
blems01/X/Overview.html">Overview.html</a>Childtree for NOUN to
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X/ch
ildtree_NOUN.html
jjjjjjjjjjjjjjjjjjjjjjLeafnodes for NOUN
Childtree for FAILURE to
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X/ch
ildtree_FAILURE.html
jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjLeafnodes for FAILURE

Flamencoing in
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X/fl
amenco

Unknowns report to
/user6/httpd/htdocs/projects/reconciler/shortProblems/shortProblems01/X/sc
ratch/warns.html
Finished shortProblems shortProblems01 X at Wed 04/27/2011 11:43:28
```

You will also see some minor warning messages - these occur when the parser or the tagger is having difficulty with unusual or malformed sentences. You may ignore those warnings.

Toward the end of the run, it tells you where your flamenco files are written. In the standard configuration, this would be:

```
/var/www/html/reconciler/shortProblems/shortProblems01/X/flamenco
```

Note this; you will use it when building the Flamenco+ database.

Examining the STAT Output

Open the overview page. If you are in the standard configuration, that will be:

- http://localhost/reconciler/shortProblems/shortProblems01/X/Overview.html

If you are running at Johnson Space Center (JSC), that will be:

- http://tommy.jsc.nasa.gov/projects/reconciler/shortProblems/shortProblems01/X/Overview.html

Look at the charts on this page; note how they correspond to your spec.
At the top of the Overview page is a small table with hyperlinks to the Failure and Noun ontologies:

Trend-Group Page
FAILURE Ontology
NOUN Ontology

Follow the link to the Failure ontology. Scroll down to *2.3.1.2 - Nonconforming_Object.*

No.	Concept	Counts within Columns
		Description
2.3.1.2	Nonconforming_Object	6
	NOT AS REQUIRE	2
	NOT MATCH	1
	NONCOMPLIANT	3

This shows that the concept *Nonconforming_Object* was evoked six times, by three different phrases, with counts as shown. The '6' is a hyperlink pointing to the LeafNode page for *Noncomforming_Object.*

Scroll to the bottom of this page. There is table of those records which received no failure-tags.

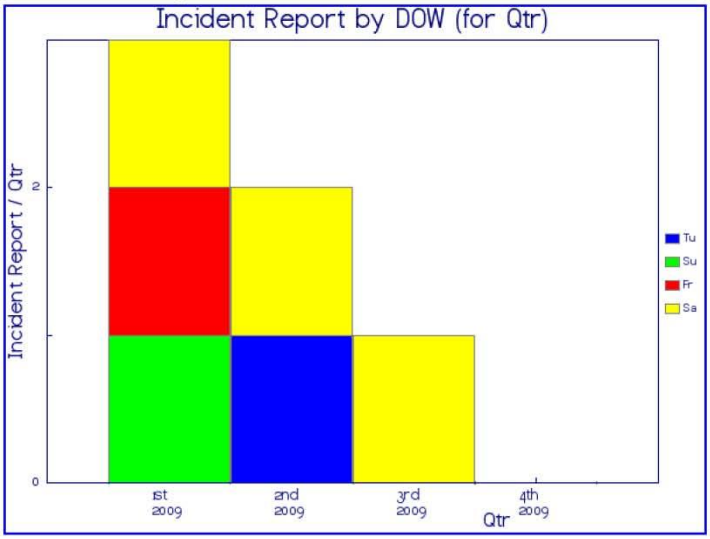
LeafNode pages

Follow the hyperlink to the LeafNode page for *Nonconforming_Object*. You will see:

Incident Report Reports mentioning any mappingword within 2.3.1.2 Nonconforming_Object

Run on Wed 04/27/2011 10:39:35

Mapping words found for Nonconforming_Object:
NONCOMPLIANT; NOT AS REQUIRE; NOT MATCH



Examples for Nonconforming_Object

1	**ID Number**: N-15	**Initiation Date**: 3/18/2009	**Tcode**: TS	**Latitude**: 335305N
	Longitude: 1171508W	**Injury**: FATL	**Crit**: 2	**Closed**: 7/4/2010
	Description: Did not match exploded drawing h0039 .			
	PartName: DECAL, IGNITER			

Compare this output to what you spec'd for *nodeArgs*.

14

Atlas Page

In the LeafNode table, for each record, there is a number in the first column. It is a hyperlink. Follow it to the Atlas page for this record. The links in these pages are not useful for the normal analysis. However, they do show interior details that will be of interest to some users. We cover them here for completeness.

ID Number	Parse Internals	UCF Parse	Description
N-15	Parse Internals	UCF Parse	Did not match exploded drawing h0039 .

You may follow the UCF Parse link to see the details of how the Stanford parser produced a parse in Treebank form, and how Fernando Gomez' MCF program apportioned it into scopes.

You may follow the Parse Internals link to see how the tags were generated within the scopes.

Creating the Flamenco Database

Go to the directory where the Flamenco+ files are. In the standard configuration, this is /usr/lib/flamenco. Your ~/.tcshrc file aliases *fla* to cd you there.

```
% fla; pwd
 /usr/local/flamenco
```

Use the bin/flamenco command to import the Flamenco+ files. If you've used the standard configuration, this will be:

```
% bin/flamenco import  \
  /var/www/html/reconciler/shortProblems/shortProblems01/X/flamenco
```

You will then be prompted for the following:
MySql server hostname: localhost
MySql server username: analyst
MySql server password: stat100 {NOT the analyst user password created when installing CENTOS.}
MySql server data base name: shortProblems

To get appropriate names for the Flamenco web page titles, your command will be:
% bin/customizeTitles shortProblems 'Short Problems Example' 'Short Problems Example' 'Small Sample Data Base'

An explanation of that format is:
% bin/customizeTitles *instanceName pageTitle pageHeading pageSubheading*

Where:

- *instanceName* must be exactly the same as in Step 1
- *pageTitle* is the web page name in the top margin of the window frame in which the web page is displayed
- *pageHeading* A name in large font on each Flamenco web page
- *pageSubheading* A subheading in smaller font than the heading

If any changes are made, it is important to execute both commands above (import and customTitles), in order, for the changes to take effect.

At this point, you should be able to view Flamenco+ results in your browser at:
http://localhost/cgi-bin/flamenco.cgi

The documents *Creating Flamenco Databases* and *STAT Flamenco User Guide* give further instructions on building other data sets and exploring the data sets you have built.

Appendix C. STAT User Guide

STAT User Guide

David Throop

Version 1.05
January 4, 2012

Forward: STAT (*Semantic Text Analysis Toolkit)* is software for performing trending analysis on data sets – data in which critical information is found within data-fields of English-language sentences. STAT works especially well with various sorts of trouble reports (discrepancy reports, incident reports, PRACA…) drawn from aerospace domains.

STAT works by parsing the sentences – into *subject / action / object / prep-phrase etc.* components. These parsed sentences are then tagged for failure-concepts, which are then graphed *vs.* time.

1 Installing and testing STAT

STAT / Flamenco+ is an advanced research prototype. It is already in use, tagging reports and producing trending graphs. However, it is not yet a fully supported tool. It incorporates several utility programs which depend on a specific operating environment. The utilities require nearly-recent versions of Perl, Python, and SBCL (Steele Base Common Lisp) which are not the most recent versions. Therefore, we strongly suggest installing STAT / Flamenco+ under the Centos 5.4 Linux system. Centos 5.4 comes preloaded with the right Python and Perl versions and installing SBCL is straightforward in this environment.

You may run Centos 5.4 as the only OS on a dedicated computer or as one of the OSs on a dual-boot machine,.or Centos 5.4 can be loaded within Oracle VM VirtualBox. Alternatively, if you are handy with installations, you may install the nearly recent versions of SBCL and Python under another Linux version and adjust paths accordingly. (Let us know how it goes.)

The following installation guide assumes you have already installed a Centos 5.4 environment and that you can login as *root*. The example also assumes a 32-bit machine, under the tcsh shell.

1.1 Installing Centos 5.4

This describes the process for installing Centos on a dual boot machine.

Go to http://isoredirect.centos.org/centos/5/isos/x86_64 and select one of the mirrors. In this example, we choose usc.edu. Navigate to http://mirrors.usc.edu/pub/linux/distributions/centos/5.4/isos/. Choose i386, download CentOS-5.4-i386-netinstall.iso (8.9 M) and burn it to a CD or DVD. This downloads an iso file – a wizard for installing Centos from the web.

The machine receiving Centos should be connected to Ethernet, rather than being wirelessly connected. Put the CD into the CD tray of the machine, and reboot. The Centos wizard will come on screen. Take the default on several screens.

- On the first screen, choose *Upgrade in Graphical Mode*.
- You will presumably choose *English* as the language and *US* as the keyboard.
- On the *Installation Method* screen, it asks on which media to find Centos; choose `http`. (Hit right arrow key to highlight <OK>, then hit <Enter>)
- As of May 2011, at most NASA sites at least, you should *not* enable IPv6 support. (To unselect IPv6, position cursor, then hit <space bar>, then <enter>)
    - If it hangs at this screen, or keeps returning to this screen, check your Ethernet connection.
- Next, it asks you for a mirror. Use the same mirror, At
    - MIRROR: *mirrors.usc.edu* (<tab> for next line)
    - DIRECTORY: /pub/linux/distributions/centos/5.4/os/i386 (<tab> to OK)
- Will take a few minutes to load.
- <spacebar> for "next"
- Choose *Install Centos,* instead of *Upgrade an Existing Installation.*
- If Linux was previously installed on your machine, choose *Remove Linux partitions and create default layout.*
    - It will want confirmation.
    - Otherwise, if enough free space is available, select *Use free space on selected drives*.
    - Otherwise, select *Remove all partitions on selected drives and create default layout*.
        - Accept defaults on next screen.
        - User GRUB boot loader (default); accept other defaults on this screen.
- In most cases, you will want to set the hostname through DHCP.
- Enter your time zone (e.g., America-Chicago for Central).
- During the installation, you will be prompted to create a root account and to create a root password. Supply one; record your choice.
- At the 'default installation' screen, accept Gnome and leave the others blank.
- The installation runs about an hour. When it finishes, remove the CD and reboot. During reboot, ignore messages about the crash kernel.
- After the reboot, it will prompt you for firewall settings; disable the firewall.
- Set *selinux* to *Permissive*.
- Near the end of the installation, the wizard will prompt you to create a user account. Choose the user-name *analyst* and a password. Record your choice.
    - User name *analyst* is case-sensitive! Installation scripts expect there to be an *~analyst/* directory.
- When you receive the login prompt, login as *analyst.* Open a terminal window by mousing-right on the background and choosing *Terminal.*

1.1.1 Utilities which come with Centos

The following needed utilities arrive with Centos 5.4:

- Python
- Perl
- Aspell
- Yum
- Firefox

1.2 Installing STAT, Flamenco+, and their utilities

1.2.1 Positioning the Distribution Files into the Right Place

Login as *analyst* (performed at the end of 1.1). The STAT distribution includes the following files:

```
STAT.sh
delete_stat.pl
fetch_stat.pl
stat_FlamencoPlus.tar.gz
stat_analyst.tar.gz
stat_cgi.tar.gz
stat_source.tar.gz
stat_source_public.tar.gz
stat_ucf.tar.gz
unpackingScripts.tar.gz
```

There are several ways to position the files.

1.2.1.1 Position files from a distribution disk

Place the distribution disk into the CD reader. Copy all of these files to the `/tmp` directory. You may either use the Gnome filesystem tools, or you may use the command line. If using the command line, the distribution disk will be under the directory /media.

1.2.1.2 Position files using the *fetch_stat* script

If you are at JSC or VPN'ed to JSC, you may get the *fetch_stat.pl* script and execute it; it will fetch the other files to the /tmp directory:

```
% cd /tmp
% wget tommy.jsc.nasa.gov/projects/reconciler/tar/stat/fetch_stat.pl
% perl -w /tmp/fetch_stat.pl
```

1.2.1.3 Position files from the webpage

Or you may download them individually from the http://tommy.jsc.nasa.gov/projects/reconciler/tar/stat directory. You can make this easier by going into Firefox 3.0.12 (the version that comes with Centos 5.4), navigate as *Edit > Preferences >Main* and setting '*Save Files to:*' to /tmp. Then click on every file in the directory.

1.2.2 Enabling Services, Permissions

Under the *System* tab, open *Services*; you will be prompted for the root password. A menu appears. Check the box next to the service *httpd* and Restart it.

Under the *Administration* tab, choose *Security Level and Firewall.* A menu appears. On the *Firewall Options* tab, set *Firewall: Disabled.* On the *SELinux Tab*, set *SELinux Setting: Permissive*.

1.2.3 Running the installation script

Once the files are in /tmp, become root and run the STAT.sh script.

You may want to make the terminal wider for easier reading of the output.

```
% cd /tmp
% su
% tcsh
% sh STAT.sh |& tee MyLog.txt
```

`STAT.sh` is a shell script. The first thing it does is unpack `unpackingScripts.tar.gz` – a tarball of six other shell scripts. Then it executes each of them. The `|& tee` causes the output (including STDERR) to both be sent to the terminal and to be copied to `MyLog.txt`. This allows you to view the output later, if you suspect something has gone wrong.

Early on in the terminal window, you will see the prompt "Press enter to continue…" Press <Enter>.

The Java installation will open Firefox, asking you to register as a JDK user. Just minimize the browser and ignore it.

Later, the CPAN installation of Perl scripts will ask you if it's OK to configure automatically. The default is 'y'; just hitting <return> is enough.

Otherwise, the script should run with no prompts.

1.2.4 Logout and Login

Once STAT.sh finishes, exit from root. You will need to type "exit" twice (once to exit tcsh, and another to exit "root"). The "whoami" command should now give a result of "analyst." Under the *System* tab, choose *Log Out Analyst.* Once logged out, log back in as *analyst.*

Why is it necessary to logout and then login again? The installation made changes in the *analyst* account, including changing *analyst's* login shell to `tcsh`. The file `~analyst/.tcshrc` sets the PATH, other environment variables, and several useful aliases. Logging in again puts you in the right shell, with the right environment.

1.3 Utility tests

To make sure that the utilities are installed and running, check their versions as follows:

MySql: The command line prompt and response:

```
% mysql -V
Mysql Ver 14.12 Distrib 5.0.77, for Redhat-linux-gnu (i686)
```

Perl: The command line prompt and response:

```
% perl -v
This is perl, v5.8.8 built for i386-linux-thread-multi
```

(The Perl portions of STAT will also run on Perl 5.10.0 and 5.12.3.)

Python: The command line prompt and response:

```
% python -V
Python 2.4.3
```

ASpell: The command line prompt and response:

```
% aspell -v
@(#) International Ispell Version 3.1.20 (but really Aspell
0.60.3)
```

Apache: The command line prompt and response:

```
% /usr/sbin/httpd -v
Server version: Apache/2.2.3
```

1.4 Starting the Parser

The UCF parser runs a separate process. Run it in its own window. Open a new terminal window. Connect to the directory where the UCF code is installed. In the Centos 5.4 configuration, this is in /usr/ lib/perl5/5.8.8/UCF_NLP-stanford. For *analyst,* the alias 'ucf' connects to that directory. Type:

```
%ucf
% cd lisp
% ./setupsystem.sh
```

Much information will scroll by. Eventually, the process prints a line starting with `(:ABSOLUTE` and pauses. The STAT code and the ucf parser communicate through socket (located in `/tmp/socket`). When STAT passes text to the parser, while it is processing the text, it prints messages to this terminal window. You may minimize this window, but don't close it.

2 Running STAT Example and Test Files

STAT may be invoked in several different modes. If a data set includes a date in each record, STAT can build trend charts showing the trends over time. However, if a dataset includes no dates, STAT can still tag the records, and produce output showing the tags.

2.1 STAT Task Description

Stat performs its tasks in one general way, with many variations:

- It reads in the source file(s), which contain English text.
  - Text sources may be from Microsoft Excel® spreadsheets, Microsoft Word® documents, PDF documents, or from a database.
  - It loads its own knowledge bases along with the sources.
- It parses the sentences in the text, then tags portions of the text.
  - Most often, tags mark words and phrases which denote a problem: *misaligned, failed to open, no authorization*.

- Verbs (*put, transfers, receives, connects)* and equipment (*camera, valve, personal parachute assembly)* can also be tagged.
- It generates graphs to visualize the tags.
  - If the data records contain dates, these graphs show trends in the data.
- It outputs the tagged data in computer-readable form, for use by other visualization / exploration tools.
  - This output can be as .tsv or .csv files (for reading into databases, spreadsheets, and many other tools) or as .xml files.
  - Output to Flamenco+ is discussed in Section 3.2.

2.2 Text Records with No Dates

STAT is shipped with several example files where each record comprises only two fields: a record-ID and some text; (some have a third field, *comment,* which explains what this record illustrates). These can be parsed and tagged, and the output files may be examined.

Similarly, if you have a data set which you want tagged, but not trended, you may copy the format of these files and run them.

The example files are stored below the ontology/source. In the Centos 5.4 distribution, this is /var/www/html/ontology/sources. As the analyst, you may cd to that directory with the alias `osource`.

```
% osource
```

The example files are below this, in `test/examples`.

Suppose you want to run the example file *forTheManual.tsv*. In a new terminal window, from the base STAT directory:

```
% Stat
% perl –w ucf/charp.pl forTheManual –verbose –excelReport
```

You can view the output from the Firefox browser at:
http://localhost/reconciler/charp/forTheManual/X/

STAT looks through the sources/test/examples directory looking for a file named forTheManual.txt or forTheManual.tsv. It reads the file with the following rules:

- Lines starting with '#' are comments and are ignored. Blank lines are ignored.
- The first (nonblank, noncomment) line holds the column headers.
- The first column is an ID field and the second column holds English text. An optional third column holds a Comment which will be included in the output.

The *forTheManual.tsv* file begins as:

ID	Sentence
# This file contains example sentences for the User's Manual	
3	The analyzer has exceeded its limited life.
4	The fastener does not have enough running torque.
9	The valve failed to open at the set pressure.
29	Prelim inspection performed without adequate qa representation.
31	Assembly does not have adequate lot traceability.
46	The SVG no longer automatically powers on when power is applied from the power supply.
124	Subsequently the unit provided stable readings and the problem was unable to be reproduced.

Because STAT was invoked with the –verbose keyword, it prints a message to the screen telling where the output files are written.

2.3 Viewing Outputs

Most of STAT's outputs are written as html web-pages.

2.4 *Directory of outputs: The Atlas file*

If we look at the Atlas file for this run, we see:

	ID	Parse Internals	UCF Parse	Sentence	Problems	Category Tags
1	003	Parse Internals	UCF Parse	The analyzer has exceeded its limited life .	UP LIMIT LIFE	Expired
2	004	Parse Internals	UCF Parse	The fastener does not have enough running torque .	DOWN RUN TORQUE	Insufficient_ Mechanical_E
3	009	Parse Internals	UCF Parse	The valve failed to open at the set pressure .	FAIL TO OPEN	Did_Not_Control_ Opening
4	029	Parse Internals	UCF Parse	Prelim inspection performed without adequate qa representation .	BAD REPRESENT	Misinforming
5	031	Parse Internals	UCF Parse	Assembly does not have adequate lot traceability .	NEGATIVE TRACEABILITY	Object_ Disorganized Traceability_Error
6	046	Parse Internals	UCF Parse	The SVG no longer automatically powers on when power is applied from the power supply .	NEGATIVE POWER ON	Failed_Start
7	124	Parse Internals	UCF Parse	Subsequently the unit provided stable readings and the problem was unable to be reproduced .	NEGATIVE REPRODUCE	Object_Not_ Testable

The table shows the ID for each record, and shows how it was tagged, in red. It also shows what mapping phrase was matched (in the *Problems* column) and what ontology concepts that mapping phrase is associated with (in the *Category Tags* column).

The two columns *Parse Internals* and *UCF Parse* are hyperlinks showing the internal details of how the sentence was tagged and parsed.

For large outputs, the Atlas information is broken across multiple files.

2.5 *Tags in Machine Readable Form*

STAT was called with the –excelReport keyword, and printed a tsv file, suitable for reading into a spreadsheet program, such as Microsoft Excel®. It produced this output:

ID	Sentence	FaultNodesEvoked
3	The analyzer has exceeded its limited life.	Expired
4	The fastener does not have enough running torque.	Insufficient_Mechanical_E
9	The valve failed to open at the set pressure.	Did_Not_Control_Opening
29	Prelim inspection performed without adequate qa representation.	Misinforming
31	Assembly does not have adequate lot traceability.	Object_Disorganized; Traceability_Error
46	The SVG no longer automatically powers on when power is applied from the power supply.	Failed_Start
124	Subsequently the unit provided stable readings and the problem was unable to be reproduced.	Object_Not_Testable

This output is suitable for reading into data mining tools.

3 Trending with your own data

3.1 Concepts: the Proj, the Case, and the Iter

You will be running different sets of data and running them multiple times. STAT writes output to different directories for these runs. Consider these possibilities:

- You have data sets from different projects, which have different formats (different numbers and names of the columns in your input.) Pass a different name for each one of these, call this the ***proj***.
  - You will need to write one specification file (specifying the data format) for each proj.
- You have multiple data sets with the same format, such as data sets for different years. Call this the ***case***.
- You may run the same data multiple times, such as rerunning data after a new version of the ontology is released, or after a set of patches to STAT has been installed. You will want to compare the old output to the new. The new should not overwrite the old. Each run gets a separate name, call this the ***iter***.

STAT stores its output within the subdirectory *$base/$proj/$case/$iter,* where $base is held in $runSwitches{directories}{base}.

3.2 Specifying your format

Consider an example case where we have a project called *shortProblems,* and a data files for that project called *shortProblems_01.tsv, shortProblems_02.tsv*. The shortProblems project has one spec file for all the data sets. It has the same base name as the project, *shortProblems.pm,* shown below.

```
use vars qw(%runSwitches);

$runSwitches{proj}{shortProblems} =
   {
    AtlasFields         => ['Sentence'],
    ID_Field            => 'ID number', # The key field for the data
    attrByQtr => [qw(Tcode)],
    columnHeaders       => [qw(Tcode Date)],
    dateField           => 'Date',     # The field holding the date for trending.
    frontFunctions => [\&nodeCharts],
    maxStripeCount => 15,
    nodeArgs            => {topcolumns  => [qw(Tcode)]},
    overviewArgs  => {attrsForTop_N_Charts => ['Tcode']},
    printCharts   => 1,
    printLeafnodes      => 1,
    reportsAre          => 'Incident Report',
    tagFieldHash  => {FAILURE => {Sentence => 1}},
    topcolumns => [qw(Tcode)],
    x_spanParamPhrase =>  'Qtr',
    nodeArgs => {long_fields  => ['Sentence'],
             short_fields  => [qw(Tcode Date)],
             catMap_lineFn => \&H_catMapWdIn,
             columnHeaders => [qw(Tcode Date)]},

    startWithNode => {FAILURE =>
                    [qw(Impaired_Controllability Incompatible Ineffective
                        Mechanically_Impaired Input_Output_Deviation
                        Agent_Deviation_or_Error Functional_Deviation_or_E
                        Process_Deviation_or_Erro Resource_Use_Deviation
                        Artifact_Problem Not_Robust
                        Damaged_or_Injured_or_Des Damage_or_Impairment_Sour
                        Object_Conformity_Problem)]});
```

Let's discuss this input spec.

- *AtlasFields* specifies which of the fields are printed out as columns in the Atlas (see above.) Fields which have been tagged will display with colored highlights.
- *reportsAre* specifies how the records will be labeled (in titles, on the axes of graphs, etc.)
- *printLeafNode* – the run will tag many concepts in the ontology (but not nearly all of them.) This switch says to print out a page for each tagged concept, showing how it trended over time and listing the records which received the tag.
- *printCharts* – For each figure in the concept reports, a .gif file is built. This is time / space intensive. For runs for checking out other aspects of the program, if *printCharts->0,* the graphs are not generated.
- *tagFieldHash* – this is a Perl HoH
    - the external key is an ontology name (usually FAILURE, sometimes NOUN, could also be VERB or PROPERTIES.
    - The internal key is the name of a field in the data

  Together, this specifies that the *Sentence* field should be tagged for the FAILURE ontology.

- *startWithNode* – In this project, we do not wish to tag very general (and uninformative) problem-concepts. We also want to exclude problem-types which are far removed from the domain (e.g., the word *ANGER* is in the problem ontology, but the few times it has been tagged on our data, it was on the text "*heat-exch anger*".) This switch says to only tag problems that occur in these nodes in the problem-hierarchy, or in their descendent nodes.

STAT recognizes many other switches, but these are enough to generate basic output.

3.2.1 Spec for Flamenco+ outputs

STAT creates files which serve as inputs to Flamenco+. Some of the information needed for Flamenco+ was already specified (e.g. the `dateField`).

Every field which STAT tagged will appear as a facet in the flamenco output. Additionally, you may specify that other fields appear as facets. It is useful to display those fields which only contain a few different values (say, 2 to 12 different values) as facets. The fields listed in `facets_not_tagged` are displayed as facets

In Flamenco+, your browsing eventually takes you to individual records, (*aka The End Game)*. In the Flamenco+ spec, `attrFields` controls which fields (from your original input) the record displays. The `attrFields` lists the fields which appear in the records. Do not list the `dateField` in `attrFields`; it will be included automatically

The STAT records, in general, are written into subdirectories of $`base/<proj>/<case>/<iter>/`. The subdirectory for the flamenco files is normally specified as '`flamenco`.'

Flamenco+ includes a keyword-search feature. It builds up an index of tokens for each record; by default it gathers those records from every field. The `token_field_exclude` property lists fields for which tokens are *not* gathered. (This capability was added when we were doing specialized searches for sharp objects, and the name of Mr. Sharp appeared frequently in the `Initiator` field.)

Facet hierarchies may be built up in complex ways. STAT's `specialFlamencoFns` spec provides a way for you to spec a function, which will produce an additional facet:

```
flamencoFns => { …
               specialFlamencoFns =>[\&files2for_CountryStateCity]…}
```

This code, for the FAA example, pulls data from three different input columns (*country, state, city)* and builds a single, 3-level facet. Listing all the ways one might build a facet from complex data are beyond the current scope of this *User's Guide*. However, the code for the `=>[\&files2for_CountryStateCity` function is a useful template.

3.2.2 Flamenco Specification File

After STAT has written the data files (the .tsv files) for Flamenco+, it writes a spec file - specifications.py. This controls the presentation of the data: What the title, subtitle, and headers should say, how attributes should be labeled, and what options should be added. The content of specifications.py

is generated from the Flamenco spec with the STAT spec file. The code below shows the spec for the Janus example:

```
flamencoFns => { …
  spec =>   # For what goes in the flamenco/specifications.py file.
         {PAGE_TITLE => 'FAA 2007 Incident Reports',
          PAGE_HEADING => 'FAA 2007 Incident Reports',
          PAGE_SUBHEADING => 'Tagged Using Aerospace Ontology Version 1.07
                              from Protege',
             HAS_ITEM_TABLE_BUTTON => 'True',
             ITEM_TABLE_ATTRIBUTES =>
               ['Event ID', 'Date', 'Narrative Failures', 'Cause Failures',
               'Narrative Equipment']}}},
```

The fields are mostly self-explanatory; for more see the Flamenco+ User's manual.

3.3 The Data File

Stat reads data specifications from a Perl file. *ShortProblem's* first data set, *shortProbelm_01.tsv*, is as shown:

ID number	Tcode	Date	Sentence
1	TS	1/1/2009	He the inserts are broke down; requested by Mark Flood.
2	FH	1/5/2009	The unclean hard drive, S/N 12312, in the control rm was detonated by absent informat
3	FH	1/2/2009	Several data errors had been exacerbated with extreme cooling.
4	FH	9/10/2009	Channel A12 on system 41.Ax is fluctuating, burst, incorrect and insufficiently dried
5	FH	10/14/2009	The coax cable was singed in a blaze. With coax cable blazing buffer overflow and tw
6	FH	6/26/2009	The clean temp sensor had debonded and was signently unmaintainable with glove on 5 f
7	FH	9/14/2009	The MBAR code will be done on DR a40630022 (p406- bearing going bad) to relief valve,
8	FA	4/21/2009	At 1950 HRS on 10/27/2006, the 1/4 ft SSATA chambers were leaking w/ accompanying noi
9	FA	6/4/2009	Power supply has bare metal spots caused by clamping for match drilling in cntrl room
10	FA	9/17/2009	2) Other components affected are relief valves: RV-GC-1, RV17
11	FA	8/28/2009	The impact of IM&TE out-of-tolerance data on the validity of past bouncing test uncle
12	FA	12/16/2009	1 FT. out-of-cal 27.2 lbs missing reqts 9 in by 12' at -9 millivolts

The tab-separated column has four columns:

1. *ID Number,* a unique identifier for the record. Each data set must have a field holding a unique record identifier (that is, a key field), though it does not have to be the first column.
2. *Tcode*, the values from this field are drawn from a small number of legal values.
    a. In a typical application, there will be many fields.
3. *Date,* a field holding a date. To draw trends, the data set must have a date or a timestamp. Default format expected is mm/dd/yyyy, and leading 0's may be omitted. Other date specifications can be handled.
4. *Sentence,* the field holding the English text to be tagged.

4 Extracting Documents for Modeling

In addition to graphing trends, STAT can also extract text from a document set, parse it, and output it as XML for other analyses, such as building models. These uses generally do not use fault-ontology tagging, but they may tag the subject / action / object triples in sentences.

For the present effort, the source file is usually a Microsoft Word® document that has been stored in .txt format. The output format is XML or sometimes tab-separated values (.tsv).

Source files contain their information in hierarchical structures - perhaps in some meaningful directory structure of files, with each file having sections and subsections, which in turn contain tables with headers, rows, and cells. The Microsoft Word® encodings (.doc or .docx format) are intentionally obscure. Saving Microsoft Word® documents as .txt files is lossy, but some of the lost information can be recovered by parsing the document. This uses various techniques which we have developed as software routines.

4.1 Reconciler Document Extraction Software

Reconciler is the software suite which performs a variety of text mining functions. To use these routines, a user must specify the documents to be parsed, which parts of the document should be extracted, the format in which the extractions should be saved, and summaries of the extraction run.

This current work documents how these specifications are encoded. Currently, the user must code the spec as Perl file in a Perl Module (.pm) file. The following discussion presumes at least some knowledge of Perl syntax.

Sample Specification Code

```
$runSwitches{topArgs}{docParse} =
  {topSections => [qw(SWRequirement)],
   sections =>
    {SWRequirement => {recognizer  => qr/^(3 (?: \.0)?) \s+ (.* REQUIREMENTS ) $/xi,
                                    nameFromRec => [qw(sectionNumber Title)],
                                    secAbbr     => 'SW Reqts Sect',
                                    storeAs     => 'TopSection',
                                    subsections => [qw(DocSection)]},
     DocSection  => {recognizer => qr/^(?: \d+B)? \s? ([345] (?: \.\d+)+) \s+ (\S.*\S) /x,
                                nameFromRec => [qw(sectionNumber name)],
                                secAbbr => 'Doc Sect',
                                hasText              => 0,
                                compatability_function =>\&numAndScope,
                                subsections => [qw(Requirement)]},
     Requirement => {recognizer => qr/^ ($rqpat) \s+ (.+?) (?: Domain: \s+ (.+))? $/x,
                                nameFromRec => [qw(reqno name domain)],
                                nameFromRecMismatchOK => 1,
                                secAbbr => 'Reqt',
                                allowedAttrs =>  ['Rationale', 'Mode'],
```

$runSwitches{topArgs}{docParse}{sections} is a hash. Each key is a document section (e.g., Chapter, Appendix, Table, VerfMethodTable). Each val is a hash with the specs for that type of section. Keys may have spaces in them but debugging is easier if they don't.

4.1.1 Specifying the Hierarchy

In the table above, three levels of document hierarchy are specified – the topSections tag specifies that SWRequirement is the top level. SWRequirement 's subsections tag specifies that DocSection will be the next level down. The XML hierarchy will have the tags named the same as this hierarchy, unless there is a storeAs tag to say otherwise. This will lead to an indentured XML structure looking something like:

```
<TopSection name="Requirements" sectionNumber="3.0">
 <DocSection name="Verification Requirements" sectionNumber="3.1">
  <Requirement reqNo="GAB.2012" name="Visual Verification" domain="TBD"
   rationale="Visual indications..."/>
  <Requirement reqNo="GAB.2013" name="Verification through testing" domain="TBD"
   mode="Preflight"/>
 </Docsection>
</TopSection>
```

4.1.2 Naming the Document Sections

The specification gives each different type of document section a name (in the example: *SWRequirement, DocSection, Requirement*). However, in the XML output, even though we've spec'ed two sections differently, we may want to record them similarly (e.g., we may have different specs for Appendices A and B, but want them both stored in the XML as <APPENDIX>). Also, section names are often long, providing clear mnemonics. In the Summary Reports, abbreviated names are better with tables.

- **storeAs** – Gives a different name to XML tag which stores this section's information. This is particularly useful when there are multiple sections which should be spec'ed differently but stored the same (e.g., different types of tables, with different columns, can all be stored as <Table>).
- **secAbbr** – Controls how counts of a section will appear in the Summary Table.

4.1.3 Recognizing the Beginning of a Section

Reconciler reads through the document line by line. It recognizes a line as beginning a document feature of interest - the start of a major section, the beginning of a table, a line of data in a table. It may find several pieces of information in the line, and it stores them.

- **recognizer** – a regexp which recognizes the first line of the section. May use capturing parens to catch attributes (e.g., 'name').
    - If the recognizer is 'INIT:', then the first line of the file is taken to be the beginning of this structure. There may only be one section with a recognizer of 'INIT:'.
- **nameFromRec** – a list of the attributes from the recognizer.
- **nameFromRecMismatchOK** – boolean

The number of capturing parens in the recognizer should exactly match the number of terms in the nameFromRec, or an error is reported, unless nameFromRecMismatchOK is true. In the following

example, the first line of a Requirement contains the requirement number, the name of the requirement, and, optionally, a domain.

```
recognizer => qr/^ ($rqpat) \s+ (.+?) (?: Domain: \s+ (.+))? $/x,
nameFromRec => [qw(reqno name domain)],
nameFromRecMismatchOK => 1,
```

The reqno name domain are stored as attributes of the Requirement.

- **immediateTransforms** – A function to invoke upon recognizing this document section. When set to &nextLineIsName, if the name isn't captured from the first line, grab it from the following one.

4.1.4 Additional Attributes of Sections

- **allowedAttrs** – If this == 1, any line starting with a few words, then a colon, is taken to be an attribute. If this is a listpointer, any line that matches a listmember is a new attribute.
- **is1lineSec** – If this is true, when the recognizer for this section is matched, info from the line is installed as child structure of the parent. Then processing resumes of the parent. Inter alia, this means that the following lines (e.g., of text-not-otherwise-recognized) will be attached to the parent structure.
- **noteTypes** – Add an attribute to the XML record noting its original docSection type (e.g. if a DRM_ApplTbl is stored as 'TABLE', this adds type=>"DRM_ApplTbl" to the TABLE record. This aids in allowing the consistency-reporting to navigate through the xml structure.
- **cleanAttrs** – In attributes, convert non-ASCII characters to their ASCII near-equivalents.
- **attributesFromParent** – a list of attributes to be copied from the next higher document-level into this structure (e.g., in the *Requirements* spec, attributesFromParent => [qw(sectionNumber sectionName)]) copies those attributes from the DocSection into the Requirement's XML structure.
- **attrProcessFns** – special handling when an attribute of a section is seen. Useful for handling .pdf->text files where multiple attributes have been concatenated. See hazard/Specifications/Basili.pm for examples.

4.1.5 Sentence Parsing

Reconciler can take text - sentences or nounphrases - pass them to a natural language parser, and tag the parsed sentences. These tags control those features:

- **sentenceParse** – list of the attributes of this section which should be sentence-parsed.
- **sentenceTag** – onto/field hash indicating what attributes should be tagged in which ontologies.

4.1.6 Other Section Specs

- **subsections** – list of sections which may be contained within this section.
- **hasText** – if this is true, succeeding lines which aren't otherwise recognized are appended to the 'text' attribute of this section.
- **is1lineSec** – trailing text on this line belongs to this section. Subsequent lines belong to parent.
- **processFn** – When this section is recognized, call this function to process the text.
  - The processFn relinquishes control either by changing the level (usually by a call to &upOneLevel) or by setting the doneWithProcessFn flag.
- **endPattern** – Pattern signals end of a section.

- **holdsNameForParent** – This attribute is the parent's name (e.g., when processing some FMEA documents, we don't encounter the name of the FMEA until we've processed a bunch of other text). This can signal that when we encounter the 'Failure Mode:' line, the next text is the name of the containing structure.

4.2 Specs for Tables

Sample code for specifying a table

```
SoftToReqTable
 => {nameFromRec => [qw(number name)],
    storeAs     => 'Table',  # Store this as a Table in xmlstruct
    processFn   => \&stuff_gTable,
    removeFn  => \&chewTilMatchGeneral,
    headers              => ['SW Req No', 'SW Req Name', 'Parent ID'],
    numberOfColumns  => 3,
    numberOfHeaders => 2,
    rowsAre              => {internal => 'TraceUp',
                             abbr     => 'Trc Up Rcd',
                             XML      => 'Record'},
    continueTableTest => \&genrl_ColMatch_orBlank_p,
    processRowFn => \&proc1LevelRow,
    secAbbr => 'SW to Reqt Tbl',
    colPattern => {0 => $IFrqpat,
                   2 => $rqsOrBlank},
    recognizer =>
            qr/^Table (6\-1) \- (.+ Requirement to Parent.*Trace.*)/i}
```

Pulling data out of tables is trickier than pulling it out of running text. These specs are particular to table processing. In particular, the .txt files saved from Microsoft Word® save each cell as a separate ^M-delimited line and do not reliably indicate the end-of-row.

4.2.1 Beginning the Table

Tables' beginnings are recognized the same as other document sections - with the recognizer tag. Typically, a document has several tables to be extracted and there will be a different spec (and a different recognizer) for each one.

4.2.2 Table Header Processing

The user must specify the number of columns in table. Usually there are the same number of header cells as columns. This may not be true, due to fused or split cells in the table header, or due to caption-text that can't be differentiated from a header cell. Proper handling for these cases can be specified too.

- **numberOfColumns** – How many columns (i.e., how many cells in a single row)?
- **numberOfHeaders** – How many header cells (if different than numberOfColumns). It will usually be necessary to inspect the table in the .txt file to determine this number.

- **headers** – These are the attributes under which the corresponding cells should be stored. The number of headers should match numberOfColumns (not necessarily numberOfHeaders). Spaces in the headers will be turned to underscores.

Sometimes, when working through a document set, the 'same' table in two different documents will have the same overall format, but with different numbers of header cells. In such a case, the user can specify what a good first row looks like. Header cells are chewed off until a good row is found.

- **removeFn** – This specifies a function to be called to skip past the headers. &chewTilMatchGeneral is usually used.
- **colPattern** – a hash. Keys are column positions (0 based), values are regexps. The removeFn will skip over cells until it finds a sequence of rows which match the regexps. This same colPattern is also used to recognize when incoming text no longer matches, and to signal the end of the table.
    - *Deprecated; use colTreatment* – left in for backward compatibility.
- **colTreatment** – a hash. This is a generalization of colPattern. It allows for capturing more than one attribute from a cell. It also allows specifying whether a cell may be blank (rather than forcing the user to write a pattern that can match a blank cell). Example—Given a cell in column 0 containing the text:
- 3.4.1.8 Ignition Mode

  The following spec

```
colTreatment => {0 => {pattern => qr/^(\d+ (?: \. \d+)*) \s+ (.+\S) \s* $/x,
                               attributes => [qw(Section_Number Title)],
                               blanksOK => 1}}
```

  will capture *3.4.1.8* as the Section_Number and *Ignition Mode* as the Title.

  This is also useful when two attributes are 'stacked' in a cell, with a linebreak, or when excluding extraneous characters from the captured contents

  If there are no *attributes*, the contents of the entire cell will be copied. Otherwise, the contents will be matched to the pattern. In this case, the pattern must have capturing parentheses; otherwise '1' will get recorded.

4.2.3 Processing the data rows

Once past the header text, the data is read in row by row. In the simplest case, each row is a record in the table and each cell is an attribute of the record.

- **processRowFn** – This is a function called to process a single table row. In the simplest case, where each row represents a record, use &proc1LevelRow. Sometimes, records continue over multiple rows. If this is indicated by some columns being blank, use &proc1LevelRowWDittos. If a single record may have multiple subrecords, use &process2levelrow.

4.2.3.1 Naming the Data Rows

A record of data has been read in from a table-row. What to call it? The issues are the same here as discussed in *Naming the Document Sections.*

- **rowsAre** – A hash specifying the names of the rows. It may contain three subspecs:
  - **internal** – This is the internal name for the record. If you have two tables in a document, and you want to track the number of records in each separately, give them different names. To track together, give the same name.
  - **XML** – The name in the XML record. Analogous to *storeAs* in a Section spec.
  - **abbr** – The name given in the Summary Reports. Analogous to *secAbbr* in the Section Spec.
- **subRowsAre** – analogous to *rowsAre* for indentured data structures produced by &process2levelrow; a hash with the same subspecs.

4.2.4 Finishing the Table

- **continueTableTest** –

4.2.5 Reporting Section Counts in Summary Reports

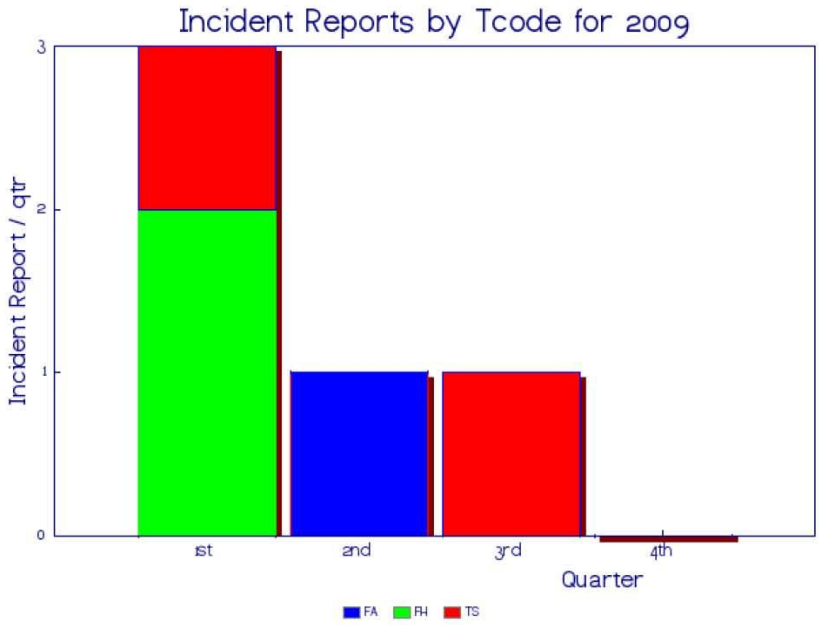
The software keeps a count (by file) of each type of document-section; (how many DocumentSections, how many Appendices...). Counts are printed out in a summary table. These tables help to quickly spot bugs – either documents that are missing information, or for document sections that aren't being recognized in some documents (often because the formatting in one file is slightly different than the others).

5 The Ontology Tree and the Concept Files

For each tagged ontology, there is a tree structure of the ontology printed out. The tree shows how many 'hits' in the data set matched each concept. For instance, this shows the tree for the example data set, around the concept of *Nonconforming Object:*

No.	Concept	Counts within Columns
		Sentence
3.3	Object_Conformity_Problem	0
3.3.1	Uncontrolled_Object	0
3.3.1.1	Lacking_Responsible_Parties	0
3.3.1.2	Nonconforming_Object	5
	NOT AS REQUIRE	2
	NOT MATCH	1
	NONCOMPLIANT	2

It shows that three different mapping phrases (*not as required, not matched, noncompliant)* were found, across five Sentences. The '5' next to *Nonconforming_Object* is a hyperlink. We follow that link to a detailed report on this concept. It starts with a bar graph, showing the trends for *Nonconforming Object.*



Remember that the spec called out:

attrByQtr => [qw(Tcode)],

This bar graph shows that, in the 1st qtr, there were three Incident Report records tagged for *Nonconforming Object.* Two of them had a Tcode of FH and one had TS.

Below that bar chart are shown the records which were tagged for *Nonconforming Object:*

1	**ID number**: 15	**Tcode**: TS	**Date**: 3/18/2009
	Sentence: Did not match exploded drawing h0039 .		
2	**ID number**: 23	**Tcode**: FA	**Date**: 6/29/2009
	Sentence: Uncompliant expired shelf life with tweezers had no intialization at 12 kg / day .		
3	**ID number**: 25	**Tcode**: TS	**Date**: 9/14/2009
	Sentence: Panels did not comply with the directive .		
4	**ID number**: 51	**Tcode**: FH	**Date**: 1/30/2009
	Sentence: The tape failed A-20 -LRB- purchase order requirements -RRB- .		
5	**ID number**: 62	**Tcode**: FH	**Date**: 3/18/2009
	Sentence: The egress blanket fails to protect crew from required temperatures and / or sharp edges .		

The format for the record follows the spec file. The key field (in this case, *ID Number)* is always shown as a short field. Two more short fields, *Tcode* and *Date,* were spec'ed. STAT will print out up to 4 short fields on a line, then add more lines. The *Sentence* field was spec'ed as a long_field, and it gets a line to itself.

Because *Sentence* is a tagged field, its tags are colored. The words that are tagged to the current concept (*Nonconforming Object*) are tagged red; words tagged to other concepts (such as *sharp edges*) are tagged green.

6 Contacting Us

For help, bug reports, or feature requests, contact:

David Throop
David.R.Throop@nasa.gov
(281) 483-5396

For questions or additions to the ontology, contact:

Jane Malin
Jane.T.Malin@nasa.gov
(281) 483-2046

Appendix D. User Guides: Maintaining and Updating the Aerospace Ontology

Maintaining & Updating the Aerospace Ontology (AO) – Abbreviated User Guide

The Ontology is always evolving in order to improve the results of their application; therefore the Ontology must be continuously maintained.

GOOD TO KNOW...

- This guide is compatible with Protégé version 4.1.0
- Double click – expands and contracts classes in the hierarchy, selects items from the drop down menu of the search engine and populates the various tabs with information about selected item
- Single click – Makes selected items active (highlighted in light blue): 'Selects' item from class hierarchies and Individual lists on left and provides information on selected item on right.
- Search engine – As typing in items to be searched, words will begin to populate in the drop down menu.... Either continue to spell complete name, or select the entry from the drop down menu and hit enter.
- To install plug-in files drag and drop files into the Plug-In folder located in the Protégé Program Folder from installation. Then restart Protégé.

GETTING STARTED

1.0
- To start Protégé double click on the protégé.exe icon:
- Select 'Open OWL ontology' option from the "Welcome to Protégé" dialogue box
- Browse for and then open the Aerospace Ontology File (file icon seen here on left)
- Once open, the Active Ontology tab view will be selected.

USER OPERATIONS - Summary

2.0
- *Exploring the Ontology*
- *Browse and Search the Ontology for a missing term*
- *Add a New Member*
- *Add a New Class*
- *Rearrange Class Hierarchy*
- *Validate and Verify Modifications*
- *Add an Object Property*
- *XLS File Setup*
- *Reading Complex Individuals*
- *Acronym Verifier Plug-In*
- *OWL Viz viewing function*
- *DL Query*
- *Export Ontology to XML*
- *Manual Operations in Protégé*

Protégé Terminology and Symbols

3.0
- *Class– A set of terms related and grouped together by a similar meaning*
- *Individuals – Members of a class*
- *Object Property – Creates relationships between Individuals*

*** EXPLORING THE ONTOLOGY TOOL AND AO CLASSES***

DESCRIPTION OF ALL THE TABS

4.0 **Here are brief descriptions of the functions of the 8 tabs in Protégé:**

- **Active Ontology Tab** – Provides information about the Ontology (i.e. metrics and annotations)
- **Entities Tab** – Provides an overview of multiple tabs, most operations can be performed in the Entities Tab using one of the sub-views.
- **Classes Tab** - Provides the view of the class hierarchy on left and class descriptions on right; where class additions/modifications occur.
- **Object Properties Tab** – Where object properties are created. (The AO currently has one object property: hasUserDefinedClassifier).
- **Data Properties Tab** – The AO does not currently utilize the Data Properties Tab. Data properties describe relationships between terms and data values.
- **Individuals Tab** - Where new terms and new members of classes get added and/or modified.
- **OWL Viz Tab** – OWL Viz allows the user to visualize the Asserted and Inferred class hierarchies using model diagrams.
- **DL Query Tab** – Allows the user to quickly and easily view information of a class (superclasses, class members, equivalent classes, etc.).
- **Excel 2 OWL Tab** – Allows the user to import Ontology additions (i.e. class, member, annotations) from a .xls file **(does not support .xlsx)**.

DESCRIPTION OF 6 MAIN CLASSES

5.0 **The following are a brief description of the 6 main classes below the class 'Thing' that form the Aerospace Ontology:**

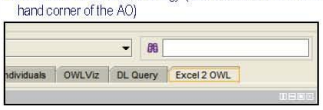
- **Acronym** – All terms that are Acronyms are a member of this class
- **Enduring** – Holds the nouns in the ontology/ provides detailed classes and mapping words for objects, descriptions, occurrences, and features/parts
- **Function**- Holds all the verbs/ classifies functions and actions for processing, placing, serving, energizing and controlling/performing
- **PROBLEM** –adjectives and nouns for entities or functions
- **Property_Value** – holds all the adjectives and adverbs
- **UserDefinedClassifier** –The members of this class are utilized by the object property hasUserDefinedClassifier. Contains names of categories that various terms relate to.

*** MAINTAINING THE ONTOLOGY – STEP-BY-STEP ***

BROWSE AND SEARCH THE ONTOLOGY

6.0 **Browse and Search the Ontology for a missing term.**

- Identify candidate for updating
- Determine if term is in Ontology (use the search tool in the upper right hand corner of the AO)



- Explore the Ontology to find a potential fit for the term (browse the Ontology class hierarchy and use the search tool)
- Research meaning of term (use dictionary & other definition sources)
- Determine if there are any missing concepts
- Determine appropriate location(s) for the term

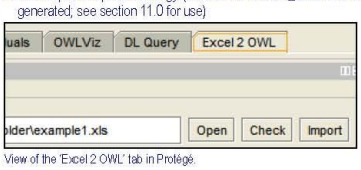
ADD A MEMBER TO A CLASS

7.0 **The next step is to add the new term to the appropriate class or classes.**

- Create and save a new XLS file that contains the member(s) additions and its new class/subclass location (*see XLS File Setup for more info*).
  - To add a term to multiple classes, add a row for each class.
  - For additions to the Acronym class: assign one member addition per row only and assign a 'isDefinedBy' annotation, see below.

	A	B	C	D	E
1	Class	Subclass	contributor	member	isDefinedBy
2	Spatial_State	Shape	J. Smith	oblong	
3	Thing			oblong	
4	Thing	Acronym	J. Smith	ET	Example_Test

- Ensure that the 'Excel 2 OWL' Tab is selected, see below figure.
- Click Open to locate the new XLS file.
- Click Check to verify class and new member existence
  - (Green – Class exists; Red – New class; Blue – New member)
- Click Import to update ontology (an XLS file named '_classifiers' is generated; see section 11.0 for use)



View of the 'Excel 2 OWL' tab in Protégé.

Maintaining & Updating the Aerospace Ontology (AO) – Abbreviated User Guide

The Ontology is always evolving in order to improve the results of their application; therefore the Ontology must be continuously maintained.

ADD A NEW CLASS

8.0 **If the need to add a new class to the Ontology arises follow these steps:**

- Create and save a new XLS file. Row 1 will contain the column headers (*see XLS File Setup for more info*).
  - Define the 'Class' in which the new class will be created
  - Enter the name of the new class below the 'Subclass' header
  - List the members of the new class
  - Add annotations (i.e. contributor, comments, etc.)

	A	B	C	D	E	F
1	Class	Subclass	contributor	comment	member	member
2	Shape_Property	New_Class	J. Smith	new class added	member1	member2
3						

- Ensure that the 'Excel 2 OWL' Tab is selected
- Click Open to locate the new XLS file.
- Click Check to verify class and new member existence
  - (Green – Class exists; Red – New class; Blue – New member)
- Click Import to update ontology (an XLS file named '_classifiers' is generated; see section 11.0 for use)
- To verify new additions search for new class (use search tool) and review modifications in the Entities Tab view.

REARRANGE CLASSES IN ONTOLOGY HIERARCHY

9.0 **After making modifications and additions to the Ontology, it may be necessary to rearrange the class hierarchy. The user may want to move a concept in the hierarchy up a level, down a level, or combine as needed.**

- Ensure that the 'Classes Tab' is selected
- Select the class that the user wants to relocate
- Use the 'Superclasses' section located under the class ' Description: ' frame to Add or Delete superclasses
  - To add a desired superclass: Select the 'Add' icon ⊕ to the right of 'Superclasses' , then select the class from the Class hierarchy tab view that the user wishes to add and then Press 'OK'
  - To delete a superclass: Select the (x) icon to the far right of the superclass under the 'Description' frame.

VALIDATE AND VERIFY MODIFICATIONS TO THE HIERARCHY

10.0 **The Reasoner can help check for any inconsistencies in the class structure. The class hierarchy that is automatically computed by the Reasoner is called the Inferred hierarchy.**

- Select 'FaCT++' from the Reasoner drop down menu (located on the toolbar).
- Verify that the Ontology is classified by selecting the Classes Tab or Entities Tab and then the Inferred hierarchy sub-tab that appears in the class hierarchy view. (Note: it might take a few seconds for the Inferred hierarchy to populate; if you see only the root class, 'Thing', the Ontology may not be classified).
- If any item is highlighted in red, it indicates that the Reasoner has found this class to be inconsistent. If any items appear in a blue color, it means that the class has been reclassified (i.e. its superclass has changed).
- You can also select Classify from the Reasoner drop down menu to classify the Ontology, but ONLY after FaCT++ has been used at least once.

ADDITIONAL TASKS

ADD AN OBJECT PROPERTY TO A NEW TERM

11.0 **New terms can be related to category descriptions from the '*UserDefined-Classifier*' class (electrical, mechanical, etc…) via an object property.**

After a file is imported into the Ontology via the Excel 2 OWL tab, a new file is generated following the naming convention of the imported file with "_classifier" appended to the end, and stored in the same location as the imported file.

ADD AN OBJECT PROPERTY TO A NEW TERM CONT.

- Navigate to the auto-generated XLS "_classifier" file name.

- Open the new XLS file.
- The XLS will contain a list of newly added members and their potential classifiers.

	A	B	C
1	Member	ObjectProperty	Member
2	oblong	hasUserDefinedClassifier	mechanical_thing
3	oblong	hasUserDefinedClassifier	software_thing
4			

- Add/Delete/Modify the list of members and classifiers. Object properties can also be modified.
- Save changes.
- Navigate to the Excel 2 OWL tab in Protégé.
- Click Open to locate the new XLS classifiers file.
- Click Check to verify class and new member existence
  - (Green – Members and object properties exist; Red – invalid entry)
- Click Import to update ontology

*** XLS File Setup ***

XLS FILE SETUP FOR E2O IMPORT

12.0 **The Excel 2 Owl (E2O) plug-in allows the user to import ontology additions including classes, members, and annotations to a class. However in order for this function to work best, the XLS file must have the proper set up. The following are tips for creating and XLS file for import to Protégé:**

- Save to an Excel 97-2003 compatible .xls file (does not support .xlsx)
- All information should be entered on "sheet1" of the XLS file.
- Row 1 will contain the column headers (Class, Subclass, Members, etc.); Order does not matter.
- The following are a list of allowed column headers for XLS file:

Class	Deprecated	Publisher
Subclass	Description	Relation
Member	Format	Rights
BackwardCompatibleWith	Identifier	SeeAlso
Comment	IncompatibleWith	Source
Contributor	isDefinedBy	Subject
Coverage	Label	Title
Creator	Language	Type
Date	PriorVersion	VersionInfo

- Class and Subclass headers must be present for import to occur.
  - If in a given row, subclass is empty, all annotations and members will be added to the specified class.
  - Annotations will apply to the lowest level class defined in a row
- The headers in row 1 can be assigned in any order, and only 1 of a column may exist for all except the Member header.
- The headers in row 1 are not case sensitive, but all items entered below the row 1 headers will be case sensitive.
- Avoid the use of spaces, #, and % signs; use underscores for spaces.

*** INDIVIDUALS WITH COMPLEX WORD EQUATIONS ***

READING AND UNDERSTANDING COMPLEX INDIVIDUALS

11.0 **Individuals can be represented as single words and as complex word-equations. Here are some tips to help understand the meaning of the complex individuals and to help create new ones when needed.**

- Lowercase terms represent Individuals
- Terms that start with a Uppercase letter represent classes
- Classes in (Parenthesis) = every individual from the class
- Classes in [Brackets] = All the classes and subclasses, along with the Individuals of its class and subclasses

Maintaining & Updating the Aerospace Ontology (AO) – Abbreviated User Guide

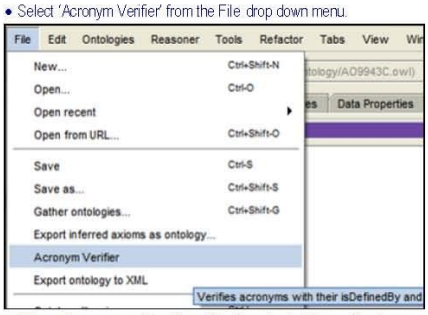
The Ontology is always evolving in order to improve the results of their application; therefore the Ontology must be continuously maintained.

HELPFUL TOOLS

ACRONYM VERIFIER

14.0 **The Acronym Verifier plug-in verifies that each acronym's isDefinedBy annotation exists as a member in the ontology; then verifies that each acronym member and its corresponding isDefinedBy member are members of the same class. Acronyms that fail either verification are exported to XLS files to be updated then imported back to the Ontology.**

- Select 'Acronym Verifier' from the File drop down menu.



- Enter a file name and location. Click Save to start the verifications.
- Upon completion, 2 XLS files will be exported to the file location:
  - XLS file #1: contains a list of acronym members missing from their corresponding isDefinedBy member's class, along with the class and subclass of the isDefinedBy member. *File naming convention: original file name*

	A	B	C
1	Class	Subclass	Member
2	Structural_Interface	Structural_Opening	A/L
3	Electrical_or_Power_Equipment	Electrical_or_Power_Converter	AMP
4	Substance	Pure_Substance	Ar
5	Obscuring	Electrical_or_Data_Noise	BER
6	Control_or_Data_Resource	Data_Signal_Resource	C3I

  - XLS file #2: contains all the acronym isDefinedBy annotations that are not yet members, along with space for the user to enter the class and subclass. *File naming convention: original file name with "_member_additions" appended to the end.*

	A	B	C
1	Class	Subclass	Member
2			AC_to_DC_converter_unit
3			ADA_Joint_Program_Office
4			ADA_development_environment
5			ADA_programming_support_environment
6			ADF_Interface_working_group

- Open XLS File#1, Verify and Save.
- Ensure 'Excel 2 OWL' Tab is selected
- Click Open, Check, and then Import.
- Open XLS file#2 next. Identify and enter the Class and Subclass for each acronym definition.
- Click Save

From here the steps are the same for when importing new data from and excel sheet.

- Ensure 'Excel 2 OWL' Tab is selected.
- Click Open to locate the new XLS file.
- Click Check to verify class and new member existence
  - (Green – Class exists; Red – New class; Blue – New member)
  - Click Import to update ontology (an XLS file named '_classifiers' is generated; see section 11.0 for use)

OWL VIZ

15.0 **OWL Viz allows the user to visualize the asserted and inferred class hierarchies using model diagrams. This tab requires installation of Graphviz software (For installation of Graphviz go to: *http://protegewiki.stanford.edu/wiki/OWLViz#Installation*)**

- To use OWL Viz, ensure that the Owl Viz plug-in is installed(the plug-in is available on the CO-ODE website at www.coode.com)
- Navigate to the 'OWL Viz' Tab
- The object selected in the model tree will be highlighted on the left column and represented with a square in the model on the right.
- A black arrow in the corner of a class means there are additional subclasses (or additional superclasses, depending on the direction of the arrow)

DL QUERY

16.0 **The DL Query allows users to quickly view information about a particular class (its subclasses, superclasses, Individuals, etc...)**

- Ensure that the 'DL Query' Tab is selected
- Prior to executing a query, verify that the Ontology has been classified, using the Reasoner (Section 10.0)
- Enter the name of a class in the 'Query (class expression)' section that you wish to query (either type it in or drag and drop from the left column).
- Select the type of information to query from the column on right (superclasses, subclasses, Individuals, etc...)
- Hit Execute and the results from the Query will be made available under 'Query results.'

EXPORT ONTOLOGY TO XML

17.0 **The Export to XML plug-in allows the user to export ontology to xml.**

- Select 'Export ontology to XML' from the File drop down menu.
- Name the file: "vers 1.07 Aerospace Ontology.xml" and specify a location to export the data to.
- Click Save to begin export.
- Enter tag names for the main Ontology classes when prompted:
  - Tag name for Acronym: ACRONYM
  - Tag name for Enduring: NOUN
  - Tag name for Function: VERB
  - Tag name for PROBLEM: FAILURE
  - Tag name for Property_Value: PROPERTIES
  - Tag name for UserDefinedClassifier: USERDEFINED

HELPFUL LINKS

18.0 **http://protege.stanford.edu**
http://www.co-ode.org

Maintaining & Updating the Aerospace Ontology (AO) – Abbreviated User Guide

The Ontology is always evolving in order to improve the results of their application; therefore the Ontology must be continuously maintained.

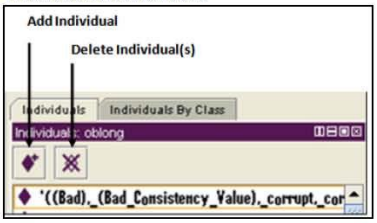
*** MANUAL OPERATIONS IN PROTÉGÉ ***

ADD A NEW TERM WITH MANUAL OPS

19.0 The next step is to add the new term to the appropriate class or classes

Add the New Term

- Switch to the 'Classes Tab'
- Select the class (from the 'Class hierarchy Tab' on left) that the new term will belong to.
- Navigate to the 'Description:' frame and select the 'Add' icon: ⊕ to the right of the 'Members' header.
- Press the 'Add Individual' button



- Type in the name of the new Individual in the dialogue box "name the Individual" and then click OK. The term should appear in the long list of Individuals.
- Click OK again and the term will be added as a member of the class.
- If the new term belongs in an additional class, select the additional class and repeat the previous 4 steps.

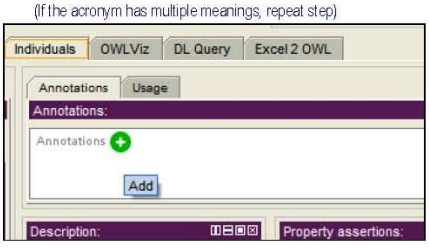
Add Annotations

- Switch to the Individuals Tab
- Select the new member/term from the column of Individuals on left
- Select the 'Add' icon: ⊕ to the right of the 'Annotations' header, (located below the Individual Annotations Tab).
- Choose the annotation type (i.e. 'contributor') from the list on left and enter the value (i.e. name of the contributor) below the Constant tab on right and then click OK.
- To add additional annotations repeat previous step
- You should see this input entered below the 'Annotations' frame.

For Acronym New Member Additions

- Select the Add icon: ⊕ to the right of the 'Annotations' header again, and the select the 'isDefinedBy' annotation from the list on left. Enter the full name of the acronym below the Constant tab on right and then click OK.

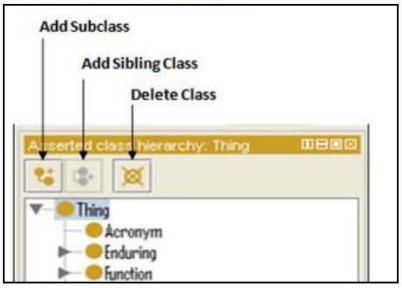
(If the acronym has multiple meanings, repeat step)



ADD A NEW CLASS WITH MANUAL OPS

20.0 If the need to add a new class to the Ontology arises follow these steps:

- Ensure that the 'Classes Tab' is selected
- Select the class in which to create the subclass (an object is selected when highlighted in light blue).
- Press the 'Add subclass' button to create a subclass of the selected class.
- Enter the name of the new class and then click OK.



ADD AN OBJECT PROPERTY TO A TERM WITH MANUAL OPS

21.0 New terms can be related to category descriptions from the '*UserDefined-Classifier*' class (electrical, mechanical, etc...) via an object property.

- Ensure that the 'Individuals Tab' is selected
- Select the Individual from the list on left that you wish to add a category description to.
- Locate the 'Property assertions:' frame (on the bottom right side of the screen), and select the 'Add' icon: ⊕ to the right of the 'Object property assertions' header
- Select the object property 'hasUserDefinedClassifier' from the left column
- Then select one of the members of the UserDefinedClassifier class from the long list of Individual on right (select either 'electrical_thing,' 'mechanical_thing,' 'process_thing,' or 'software_thing', or a new classifier category if one was identified).
- Then select OK and the object property will be added under the 'Object property assertions' view.

User Guide: Maintaining & Updating the Aerospace Ontology

User Guide: Maintaining & Updating the Aerospace Ontology

Table of Contents

1.0 Introduction

Ontologies are constantly evolving in order to expand the domain or improve the results of their application; therefore the Ontology must be continuously updated and maintained. The user of the Aerospace Ontology (AO) will find that a majority of their time with the Ontology will be spent exploring, trouble shooting, and updating the Ontology. The user guide is divided into 5 main tasks involved in maintaining and updating the Aerospace Ontology. These 5 tasks are as follows: 1) Browsing and searching, 2) Adding a new term, 3) Adding a new class, 4) Rearranging the class hierarchy, and 5) Verifying and validating modifications to the class hierarchy. Additional actions in support of these 5 tasks (adding annotations, object properties, and classifiers), will be reviewed and demonstrated.

Helpful tools and tips are provided throughout the guide to provide ease of navigation and utilization of the Aerospace Ontology. Of these helpful tools, are some unique plug-ins designed to assist the user in updating and maintaining the Aerospace Ontology. The 'Excel 2 OWL' and 'Acronym Verifier' plug-ins allow the user to make updates to the Ontology via use of an Excel spreadsheet that is imported and exported to the Ontology. Throughout the user guide, use of the plug-ins with in the 5 tasks for maintaining and updating the Ontology, will be explained and demonstrated.

1.1 Background Information

Ontology defines a hierarchical set of classes and terms that are used to describe and represent an area of knowledge. The Aerospace Ontology (AO) contains terms (words and phrases) and term relationships (similar to a thesaurus, but more detailed with word phrases, relationships, and properties). The AO is designed for identifying types of problems (mishaps, failures, anomalies, discrepancies) in the aerospace domain. Currently the AO includes a wide variety of types of problems with hardware, software, processes, paperwork, human and organizational issues. Problems are often stated as phrases that include a negative property and an object or action that has that property. Therefore, the AO includes not only classes of problems but also classes of Property Values, Functions (or actions) and Enduring things (objects, occurrences and states).

The Semantic Text Analysis Tool (STAT) uses the AO. STAT is used to extract key information from text in documents and database records using advanced parsing and ontology to interpret the meaning of problem descriptions. STAT consists of a statistical parser, an algorithm that recognizes and fills empty categories in the output of the parser to reconstruct clauses, and a semantic interpreter and tagger that uses the AO. STAT uses its methodology along with the AO to interpret English text sentences and add tags to database records containing these sentences. These sentences can be derived from documents or text fields in data records. The tagged data records can be used in applications for searching, browsing and mining the data.

2.0 Starting the Application

Protégé is available to download via the web at http://protege.Stanford.edu. The user guide is compatible with Protégé 4.1.0, which is the required version.

Note: Protégé Plug-ins are available via the CO-ODE website at http://www.co-ode.org. It is recommended to use the OWLViz plug-in, which is available from the CO-ODE web site, or can be installed when Protégé 4 is installed. For more information on OWLViz see section 12.

After installation of Protégé has completed, the application is ready for use. The following instructions will explains how to get started with the Aerospace Ontology:

1. Start the Protégé application
2. Select the option *Open OWL ontology* on the "Welcome to Protégé" dialogue box shown in Figure 1.
3. Find and open the Aerospace Ontology File.

*Note: Another way to access the AO file is to double click on the *.owl file from its saved location. This method will automatically start the Protégé application.*

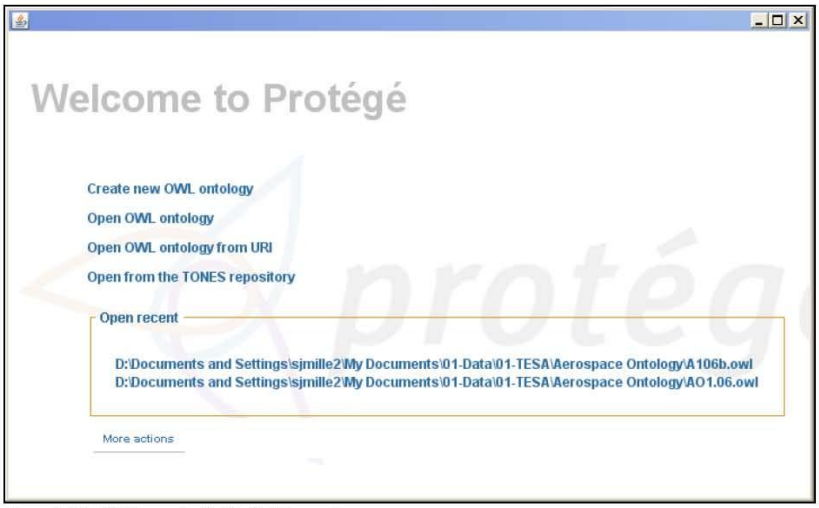


Figure 1: The "Welcome to Protégé" dialogue box

Once the Aerospace Ontology file is open, the image in Figure 2 with the **Active Ontology Tab** active is visible:

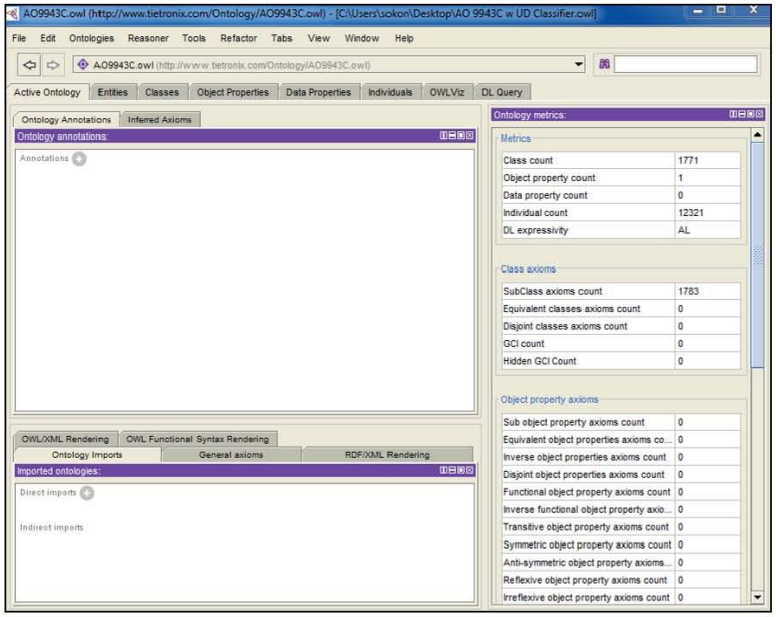


Figure 2: The AO Ontology upon starting the application with the Active Ontology Tab visible (*view of Protégé ver. 4.0.2*)

2.1 Installing Protégé Plug-Ins

Protégé Plug-ins are available for download via the CO-ODE website at http://www.co-ode.org. The STAT-AO package includes additional Tietronix plug-ins to assist the user with Ontology modifications: Excel 2 OWL plug-in, Export ontology to XML plug-in, and Acronym Verifier plug-in. The plug-ins were tested on Windows, Linux, and Mac OS X.

For more information on 'Excel 2 OWL' plug in see section 3.1. For more information on 'Acronym Verifier' plug-in see section 5.3. For more information on 'Exporting Ontology to XML' see Section 12.3.

Plug-In Installation Instructions:
Note: Tietronix Protégé plug-ins are only compatible with Protégé version 4.1.0 and have been tested with Windows, Linux, and Mac OS X.

1. Save the zipped file containing plug-ins to user's computer. Unzip the file to access plug-in files.
2. **For Windows and Linux:** Navigate to the plug-in folder located in the Protégé 4.1.0 program folder created during installation. Move the three plug-in files into that plug-in folder.

For Mac OS X: Open Finder | Navigate to the Protégé-4.1.0 installation folder (default installation is /Applications) | Click 'Go to Folder' in the Go Menu | Paste the following path to access the plug-in directory: *Protégé-4.1.app/Contents/Resources/Java/plugins* | Move the three plug-in files into that plug-in folder.

3. Start Protégé application. If Protégé was running, re-start the program.

4. Verify success of installation. Navigate to the Tabs menu in the Protégé toolbar and select (check) 'Excel 2 OWL' from the Tabs drop down menu. The tab will become visible at the end of the list of tabs, see Figure 3. To verify installation of the 'Export ontology to XML' and 'Acronym Verifier' plug-ins, navigate to the File menu item in the toolbar, and the plug-ins should be listed as options in the File drop down menu.

3.0 Exploring the Protégé Tool and the Aerospace Ontology Classes

This section provides an overview of the Protégé tool and the Ontology classes. Section 3.1 **Error! Reference source not found.**describes the nine main tabs depicted in Figure 3 and their uses in Protégé. Section 3.2 describes the six main classes of the Aerospace Ontology.

3.1 Search and Protégé Tabs

This section provides brief descriptions of the function of each tab in Protégé. The sections that follow elaborate the functionality accomplished within the Protégé tabs. Many of these tabs provide navigation capabilities. (*See http://protegewiki.stanford.edu/wiki/Protege4GettingStarted#Navigation*). To get familiar with the Aerospace Ontology, the most frequently used tabs are the Entities Tab, Classes Tab, and Individuals Tab. The Class Hierarchy pane is accessible from all these tabs. Search is also frequently used.

Figure 3: A snapshot of the various tabs in Protégé

Active Ontology Tab – Protégé opens in this tab, which provides information about the Ontology (i.e. metrics and annotations).

Entities Tab – Supports exploration of classes, individuals and properties that are also available in other tabs. Most operations can be performed in the Entities Tab using one of the sub-views. (See Figure 5)

Search - The Find box on the upper right of each screen provides another common way to navigate and explore the ontology. The search is global across all entities in the ontology. The search menu updates as text is entered. Double click to select an item from the list.

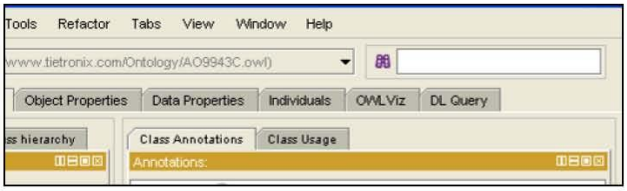


Figure 4: The Find box in the upper right corner of each screen in Protégé

Classes Tab - Provides the view of the class hierarchy on left and class descriptions on right; new classes can be manually entered and organized using either the Classes Tab or the Entities Tab (*For more information see section 6 and 13.3*)

Object Properties Tab – Where object properties are created. Object properties provide the means to relate terms in the Ontology. In the current AO there is only one object property defined: 'hasUserDefinedClassifier' (*For more information see section 9*).

Data Properties Tab – The AO does not currently utilize data properties and the Data Properties Tab. Data properties describe relationships between terms and data values. (*For more information see Appendix D)*

Individuals Tab – Where new terms (words and phrases) and new members of classes get added and modified. (*For more information see section 5 and 13.1*)

OWL Viz Tab – OWL Viz allows the user to visualize the asserted and inferred class hierarchies using model diagrams. This tab requires installation of Graphviz software (*For more information on OWL Viz see section 12.1, for installation of OWL Viz see section 2, and for installation of Graphviz see http://protegewiki.stanford.edu/wiki/OWLViz#Installation*)

DL Query Tab –Allows the user to view information, such as superclasses, class members, equivalent classes, etc. for a particular class. (*For more information see section 12.2*)

Excel 2 OWL Tab – Excel 2 OWL, a Tietronix plug-in, provides the capability of adding information to an existing ontology via importing data entered in an Excel 97-2003 compatible .xls file (*does not support .xlsx*). This tab is essential for updating the ontology using good practices for documentation and configuration control. The Excel 2 Owl Tab allows for the following:

- Add a subclass to a class
- Add members to a class
- Create a new class
- Add annotations to a class (i.e. comment, isDefinedBy, etc.)

Valid Column Headers for the Spreadsheet:		
Class	Deprecated	Publisher
Subclass	Description	Relation
Member	Format	Rights
BackwardCompatibleWith	Identifier	SeeAlso
Comment	IncompatibleWith	Source
Contributor	isDefinedBy	Subject
Coverage	Label	Title
Creator	Language	Type
Date	PriorVersion	VersionInfo

Table 1: List of column headers allowed for XLS spreadsheet for import to Ontology

Both Class and Subclass headers must be present for import to occur. If in a given row, subclass is empty, all annotations and members will be added to the specified class.

The headers in row 1 can be assigned in any order, and only 1 column may exist for all headers except the Member header. The column headers are not case sensitive, but the remaining information in the file is.

All information should be entered on "Sheet 1" of the XLS file. Avoid the use of spaces, #, and % signs. If spaces are needed, use underscores. (*For additional information see sections 5.0 and 6.0)*

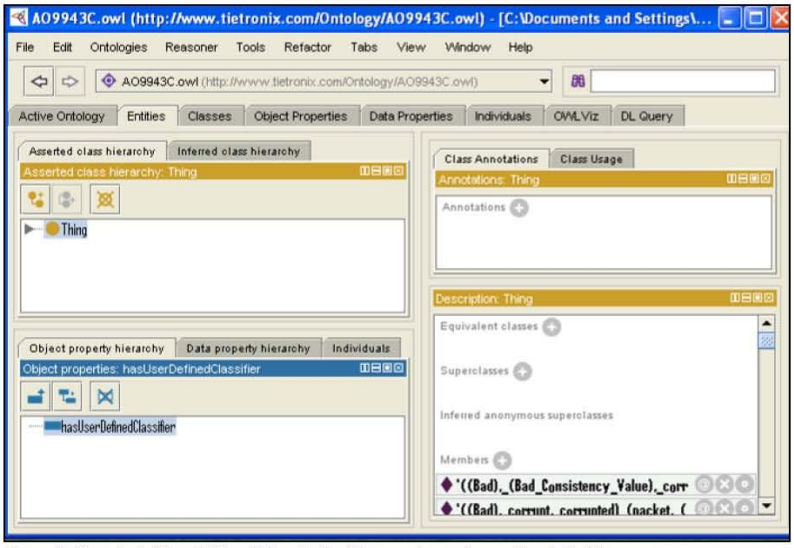


Figure 5: Snapshot of the Entities Tab, with the Classes view active on the right side.

3.2 Understanding the Aerospace Ontology Classes

In the Aerospace Ontology, Classes represent a set of terms that are related and grouped together by similar meanings. For example the class '**Fruit**' would consist of terms such as '*apple*' '*orange*', and '*peach*'. Classes in Protégé are identified with yellowish circles next to the Class name. For the remainder of this guide, Classes will be referred to in bold letters with single quotation marks. Terms that are members of a class will be italicized with single quotation marks.

The Superclass 'Thing'

The top level class of any Protégé ontology is the class '**Thing**'. All terms in the Ontology are considered to be a '**Thing**'; hence all terms in the ontology are members of the superclass '**Thing**' and all classes that are created are a subclass of '**Thing**'.

The Six Primary AO Classes

Expanding the superclass '**Thing**' (click the grey triangle by '**Thing**' one time), displays the six defined primary classes within the Aerospace Ontology, as shown in Figure 6. These classes and their subclasses contain all the terms in the Ontology. Each term is a member of at least one of the AO defined classes. These classes will become of major use when modifying or adding new terms to the Ontology.

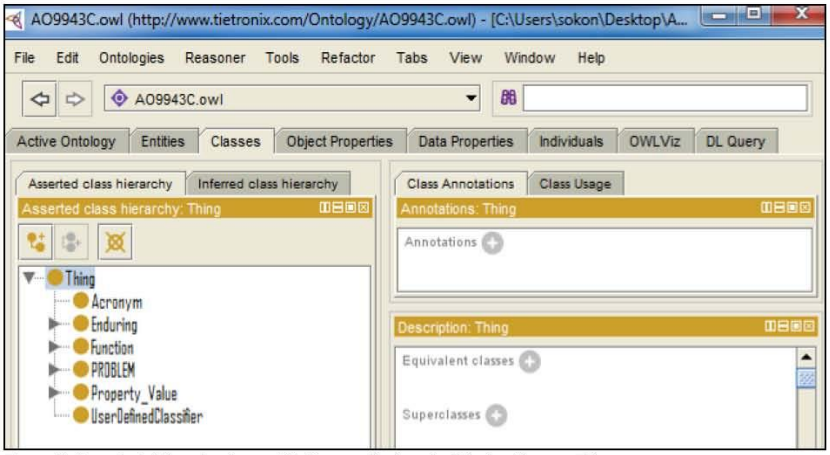


Figure 6: Snapshot of the six primary AO Classes displayed within the Classes Tab.

The following are a brief description of the AO primary classes:

1. '**Acronym**' – All items in the ontology that are Acronyms are a member of this class (*See section 5.2*)
2. '**Enduring**' – Holds the **nouns** in the ontology/ provides detailed classes and mapping words for objects, descriptions, occurrences, and features/parts (*see Appendix C*)

3. **'Function'**- Holds the **verbs**/ classifies functions and actions for processing, placing, serving, energizing and controlling/performing (*see Appendix C*)
4. **'PROBLEM'** –**adjectives and nouns** for entities or functions/ damage, hazards, impairments, failures and deficiencies, risks, symptoms and causes. (*see Appendix C)*
5. **'Property_Value'** – holds **adjectives and adverbs** (*see Appendix C*)
6. **'UserDefinedClassifier'** –This class contains the descriptive names of categories to which the various terms in the Ontology pertain. Currently these categories are: electrical, mechanical, software and process. The members of this class are utilized by the object property 'hasUserDefinedClassifier'. (*See section 9*)

Explore the AO by expanding the PROBLEM class hierarchy several levels down. When a class has associated terms, they will pop up as Members in the Description pane on the right. Do the same with the Enduring class hierarchy, the Function hierarchy and the Property_Value hierarchy of classes.

Use search to find out what classes a particular term is in. For example, type in *'filter'*. Double click on the Individual in the search menu that pops up (the option with the pink diamond in front of it). By clicking on either the Type **'Filter'** or the Type **'Separator_or_Cleaner'**, in the Description pane, you will see a new page with that class highlighted in the AO hierarchy in the pane on the left. You will also see the terms associated with this class in the Description field on the same page. Use the Protégé back button to return and inspect the other Type. You will notice that the **'Filter'** class is a **'Process'** type of **'Function'** and the **'Separator_or_Cleaner'** class is a **'Physical_Object' type of 'Enduring'.** The term *'filter'* can mean either type of thing when used in English text. The AO permits either interpretation of that term, depending on the capabilities of the parser.

For more description of the primary AO classes, see Appendix C.

4.0 Browse and Search the Ontology

This section describes the steps of the first task in maintaining the Aerospace Ontology: Browse and Search the Ontology for a missing term.

Step 1.) Identify candidate for updating

There are various reasons that can trigger the need for ontology updates, these include the following:

- Including terms from a new domain
- Incomplete search results – this could have been due to misspelling of words or missing synonyms
- Description of class content is either missing terms, or class content is mismatched, or missing relationships between terms

- When the Ontology user encounters a situation when the Ontology does not behave as intended.

Step 2.) Determine if term is in Ontology

This can be done by searching for the term in the search field in the upper right hand corner (See Figure 7)

For example, suppose there is a small taxonomy to incorporate that covers types of pilot error. One type of error is: Task Repeated, e.g. pilot presses the correct button twice. The search for *'task_repeated'* fails. Searching for *'repeated'* would locate '**Excessive_Repetition**', but it is a type of '**Property_Value**', not a type of '**PROBLEM**'. It would be necessary to search next for *'(Excessive_Repetition)'* … to see if it appears in any '**PROBLEM**' classes. (There is more about compound expressions in Section 11.) Compound terms of this type would be found in two '**PROBLEM**' classes: '**Too_Often**' and '**Overdone_Performance**'. These classes could be judged sufficient to map to Task Repeated in the pilot error taxonomy. Alternatively, the phrase *'task_repeated'* could be added to '**Excessive_Repetition**' or to '**Too_Often**'; or a new subclass, '**Task_Repeated**', could be added to '**Overdone_Performance**'. There are also other possibilities.

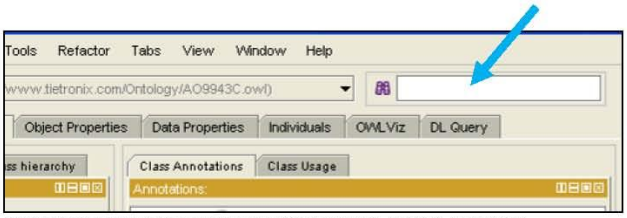


Figure 7: The search field in the upper right corner in Protégé is depicted.

Step 3.) Find where term fits in the Ontology

If the term is new to the Ontology, the user begins the process of determining where the new term will be placed. Can the term fit into one of the existing classes or will a new class be required to account for the new term?

If the term exists in the Ontology, the user begins evaluating the current classification. Is the term in the right classification, should the term belong to additional classes?

Step 4.) Research meaning of term

To determine where the term fits in the Ontology, it may be useful to find out more about the term and in what context it is used.

- Look to external sources like various websites and definition finders, i.e.: Answer.com, Wikipedia, Google places, etc.

Step 5.) Determine if there is a missing concept

After researching the meaning of the term, the user may find that the term has additional definitions not yet accounted for. This may require that the term be placed in additional classes to capture all possible meanings.

Step 6.) Determine appropriate location for the new term

Now that the user has researched the term and identified any missing concepts, it is now appropriate to determine the actual locations for the term.

Answer the following questions to determine the location: Does a new class need to be created for the term? Does the addition of the term create a need to rearrange the class hierarchy to better represent all the concepts associated with it?

4.1 Example 1: Browsing, Searching, and Identifying a New Term

The above six steps will now be demonstrated using the following example:

Step 1. Identify candidate for updating

Example Situation: The STAT tool missed tagging a document of interest because the term 'oblong' was missing. The need to add the term '***oblong***' to the Aerospace Ontology arises.

Step 2. Determine if the term ('oblong') is in Ontology

First look up 'oblong' in the search tool (watch for spelling errors)… Notice that it is not recognized by the search tool and hence does not exist in the Ontology.

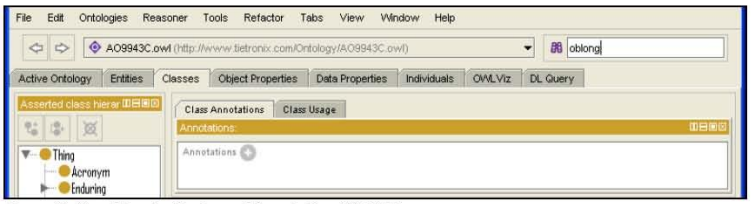


Figure 8: Searching for the term oblong in the AO Ontology

Step 3. Find where term fits in the Ontology

If 'oblong' was found to exist in the Ontology, the user will then explore the class hierarchy to determine if the term's placement is suitable or if changes are needed.

If 'oblong' was not found to exist in the Ontology, the user will begin to explore potential locations where the new term will be placed. There are multiple ways to search through the Ontology classes, and class members:

- One way to search the ontology for a potential fit is through the use of the 'Class hierarchy' (in the 'Classes Tab').
  - For example, the user can start by browsing under the class '**Property Value**', and then explore its subclasses (see Figure 9). To view the existing members of a particular class, click once on a class in the 'Class hierarchy'. This method of browsing through every class in the hierarchy can be a tedious task, although a good way to become familiar with the exiting class hierarchy.

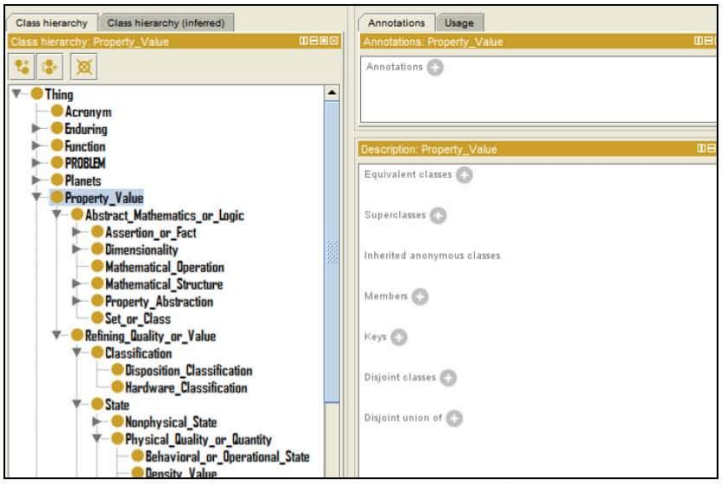


Figure 9: Searching for the best location for a new term through the Class tree rather than using the search engine can be a daunting task

- Another way to search the Ontology for potential class placements is to use the search field in the upper right hand corner, to browse for potential classes.
  - For example, since oblong is a shape, we can start by searching for the class '**Shape**'. Once selected, the user can view all members of the class '**Shape**' and proceed to explore the various members of this class (See Figure 10). Some members belong to additional classes aside from '**Shape**', for example '*curve*' and '*cylinder*', that we may want to explore for potential fits. To explore the members of a class, click once on the member. The view will then switch to the Entities Tab to provide a more detailed description of the member and its class 'Types'.

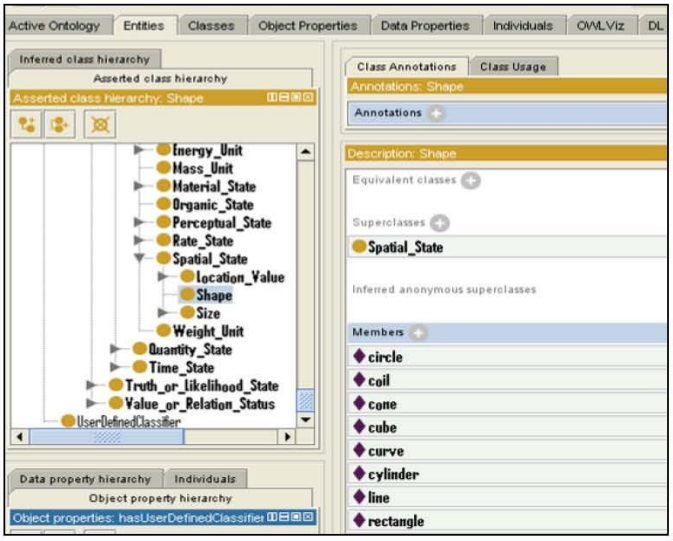


Figure 10: Selecting '**Shape**' from the search engine will make the 'Entities Tab' view active.

- The search field in the upper right hand corner of the AO Ontology can also be used to browse for similar member terms.
  - For example, the user can search for a member called '*circular*' or '*circle*' in the search tool to see if similar words to '*oblong*' exist in the Ontology, and then to explore which classes these existing terms belong to in order to determine a potential fit.

Step 4. Research the meaning of the new term, 'oblong'

Research the meaning of the new term, 'oblong', using dictionaries and other definition sources. Find out as much as possible about the term and determine if 'oblong' would be an accurate mapping word for the class '**Shape**' or any other potential fits determined in Step 3. The following are results from searching the web and Microsoft Word definitions and synonyms (some key terms are emphasized in bold):

> **Oblong** –*describing something that is longer than it is wide; having the shape of or resembling a* ***rectangle*** *or* ***ellipse****;* ***four-sided figure; quadrilateral; parallelogram; rhombus****;* ***diamond, square****; etc....*

From the definitions of oblong one can agree that the term represents a **property** that describes a **shape.** This helps us validate the assumption from Step 3 that the class '**Shape**' may be the right fit for the new term.

Step 5. Determine if there is a missing concept

Reviewing the definitions of 'oblong' obtained in Step 4; we can now determine if there are any additional missing concepts. We can determine that aside from being a shape (noun), '*oblong*' can also be used as an adjective to describe properties of a shape (i.e. *"describing something that is longer than it is wide"*). This second definition of '*oblong*' is a potential missing concept that may require '*oblong*' to be added to another class aside from '**Shape**'.

Step 6. Determine appropriate location for the new term

Now that we have explored the meaning of the new term and reviewed any missing concepts, we can now perform a more thorough search through the existing ontology to find the location(s) for the term, 'oblong'.

In Step 5, we determined that '*oblong*' has an additional concept besides a shape; the term can be used to describe properties of a geometric figure. We already explored the class '**Shape**' in Step 3, now we want to browse the Ontology for a class to fit the descriptive definition of '*oblong*'.

Navigate to the search field, and begin typing 'shape'. As you begin to type, take note of the list of terms and classes that appear in the drop down menu (*see Figure 11*). One class that appears is the class '**Shape_Property**'; let's select this class (by double-clicking) to begin our search.

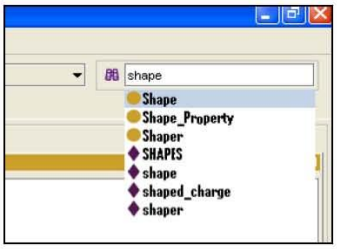


Figure 11: Searching for the term shape

Once '**Shape_Property**' is selected, a view with the description of the class will appear. Under this description is a list of all the members of the class '**Shape_Property**': *'geometry'* and '*shape'*. It appears that 'oblong' is a bit more specific then these other members.

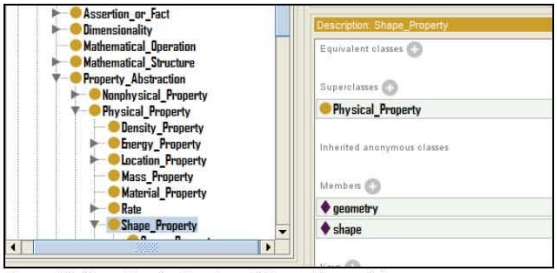


Figure 12: Searching for the class '**Shape_Property**'

At this point, we can determine that a new class is required to account for the second definition of '*oblong*'. In conclusion, the new term 'oblong' will be placed in the existing class, '**Shape**', and a new class, which will be defined and created in Section 6.0: Add a New Class. Adding a new term to the Ontology will be discussed in Section 5.0: Add a New Term.

5.0 Add a New Term

This section describes the steps of the second task in maintaining and updating the Aerospace Ontology: Adding a new term to the Ontology. Members of a class are represented by terms with purple diamonds to the left (see figure 10). In Protégé these terms are referred to as Individuals. In Protégé a group of Individuals that fall under the same category, create a class; and hence these Individuals are identified as members of a class. Referring to a class named '**Fruit**', '*apple*', '*orange*', and '*peach*' would be 3 Individuals that are considered members of the class '**Fruit**'. For this user guide, Individuals will be referred to in italicized letters with single quotation marks.

5.1 Example 2: Adding Members to a Class

The following steps demonstrate how a new term gets entered as an Individual and classified into the Ontology. This example will demonstrate adding members through use of the Excel 2 OWL tab (a function that allow the user to import additions from an excel spreadsheet). To add terms manually through Protégé see Section 13.0: Manual Operations in Protégé.

Steps to Add a New Member

1. Start a new XLS file.
2. Enter the following column headers in row 1: Class, Subclass, and member.

   Row 1 will always contain the column headers (i.e. Class, Subclass, Members, etc.); no order is necessary for column headers (see Figure 13, row 1).

   Note: Reference section 3.1, Excel 2 OWL Tab for XLS file tips

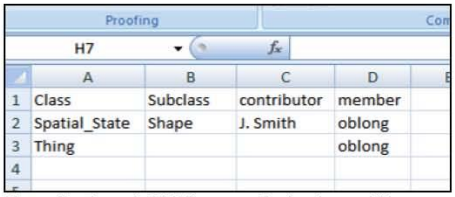


Figure 13: Example XLS file set up for Ontology additions, row 1 contains the Ontology addition headers

3. Enter the new member 'oblong' below the column member. Enter the subclass it is to be a member of, for this example: '**Shape**'. Then enter the superclass of '**Shape**', '**Spatial_State**' below the header class.

4. To add another class type for the new member 'oblong', add a new row:

   For example, to assign '*oblong*' a member of the class '**Thing**', enter a new row, with Class: Thing, and member: oblong (see Figure 13, row 3).

5. Add column headers for annotations (comments, contributor, etc.). Annotations will only be assigned to items located in the column with the subclass header (accept for the Acronym class, which will be discussed further is the following section). For a list of annotation headers supported by the Excel 2 Owl plug-in, see Section 3.1.

   Annotations provide a means for documentation; both within the spreadsheet itself and within Protégé (see Figure 14).

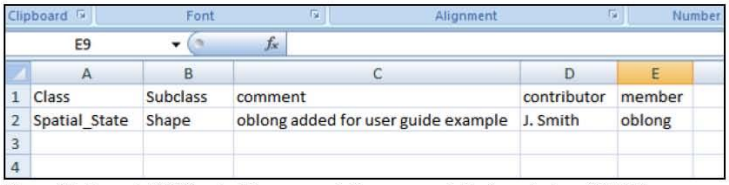


Figure 14: Example XLS file depicting an annotation, comment, for the subclass: '**Shape**'

6. Save the XLS file (does not support .xlsx).

7. Next navigate to the Excel 2 OWL tab in Protégé.

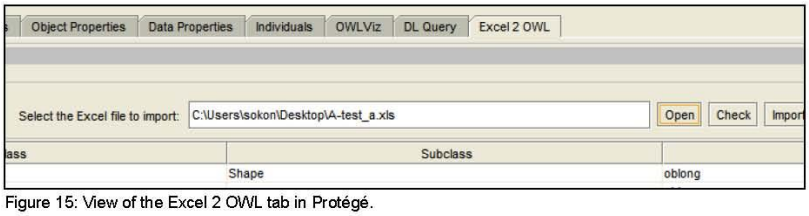


Figure 15: View of the Excel 2 OWL tab in Protégé.

8. Click Open and browse for the saved XLS file.

9. Click Check to verify class and new member existence; check will only verify classes, subclasses, and members.

   - Green cell – a class or subclass with this name exists in the Ontology.
   - Red cell – a class or subclass with this name does not exist in the Ontology. Unless the user is creating a new ontology class, both columns should be green, if not, check spelling of classes in XLS, make any necessary corrections, and repeat steps 5 – 8.
   - Blue cell – a member is new to the entire ontology. If a member is new to a class and not to the ontology, the cell will remain white.

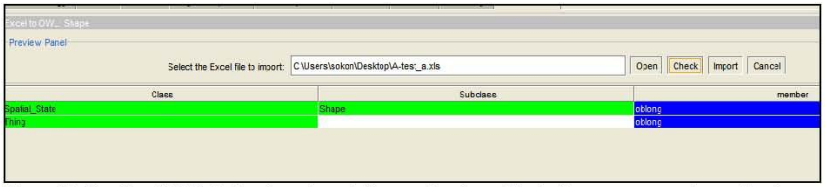


Figure 16: Checking XLS file before importing additions to Ontology. Blue indicates a new member to Ontology.

10. Click Import. Upon selecting import the Ontology is updated and another XLS file is generated named "_classifiers". Use of the auto-generated XLS file will be discussed in Section 9.3.3.

    If import is a success the user will receive a message, with a list of new additions to the ontology (if a member already existed, it will not be contained in the list).

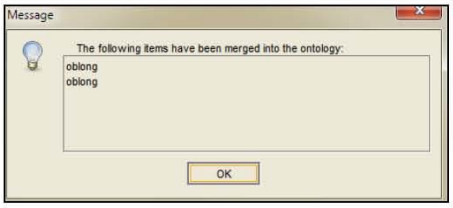


Figure 17: After importing additions, a message appears notifying user of new members and classes

11. To verify success of member '*oblong*' addition, type oblong in the search field in the upper right, and select enter. The Entities tab will populate, showing '*oblong*' as a member of class: '**Shape**' and '**Thing**':

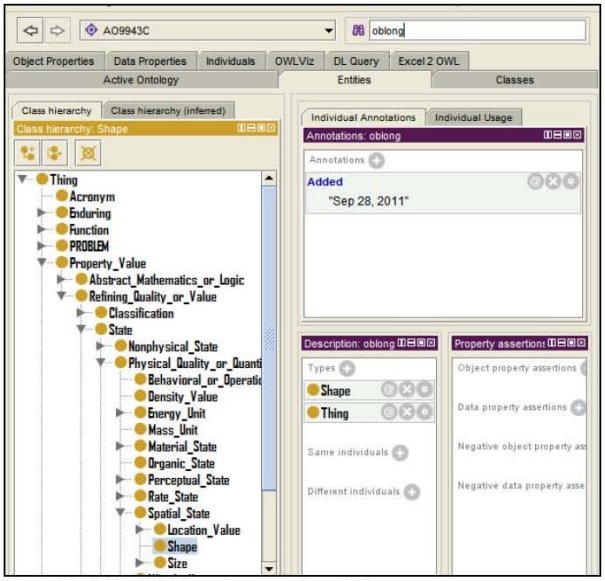


Figure 18: Entities Tab View of new member addition.

*****Additional Practice**********

For additional practice, add '*diamond*' and '*rhombus*' as Individuals to the class '**Shape**'.

The imported XLS file should appear as follows:

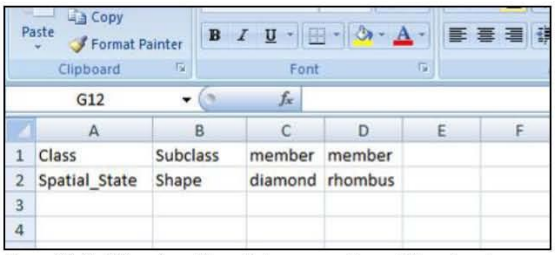


Figure 19: XLS file setup with multiple new member additions to a class.

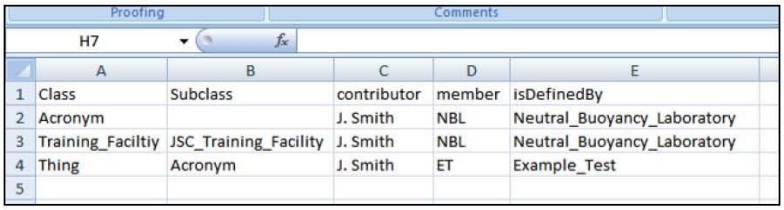
5.2 Add Members to the Acronym Class

The 'Acronym' class contains all the acronyms in the Ontology even those acronyms that are members of other classes. Adding a new member to this class is accomplished in the same way we add a new term to

any other class in the Ontology, the only difference is with acronyms, the annotation, 'isDefinedBy' will be added to a member and not the subclass. The following steps will demonstrate how adding an acronym to the Aerospace Ontology is accomplished:

Steps to Add an Acronym

1. Start a new XLS file.
2. Enter the following column headers in row 1: Class, Subclass, isDefinedBy, member, contributor, etc. (see Figure 20, row 1).



	A	B	C	D	E
1	Class	Subclass	contributor	member	isDefinedBy
2	Acronym		J. Smith	NBL	Neutral_Buoyancy_Laboratory
3	Training_Faciltiy	JSC_Training_Facility	J. Smith	NBL	Neutral_Buoyancy_Laboratory
4	Thing	Acronym	J. Smith	ET	Example_Test
5					

Figure 20: XLS file setup for Acronym class additions

Note: Reference section 3.1, Excel 2 OWL Tab for XLS file tips

3. Either enter the name 'Acronym' below class, or designate 'Acronym' as the subclass and 'Thing' as the class. Then enter the new member acronym, for example 'NBL' below the column member. Add the acronym definition (i.e. Neutral_Buoyancy_Laboratory) below the isDefinedBy header.
4. Add a new row for each additional acronym member, whether it's the same acronym assigned to a different class, or a new acronym member (unlike the other classes, Acronym class can only support one member per row). This allows for each member to contain a unique isDefinedBy annotation. See Figure 20.
5. Save the XLS file (does not support .xlsx).
6. Next navigate to the Excel 2 OWL tab in Protégé.

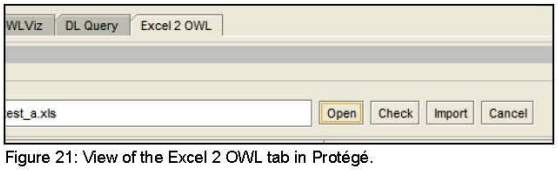


Figure 21: View of the Excel 2 OWL tab in Protégé.

7. Click Open and browse for the XLS file.
8. Click Check to verify class and new member existence; check will only verify classes, subclasses, and members.

12. Click Import. Upon selecting import the Ontology is updated and another XLS file is generated named "_classifiers". Use of the auto-generated XLS file will be discussed in Section 9.3.3.

    If the import was a success the user will receive a message, with a list of all new members added to the ontology (as a whole).

9. To verify updates, search for the acronym (i.e. '*NBL*'), and verify additions.

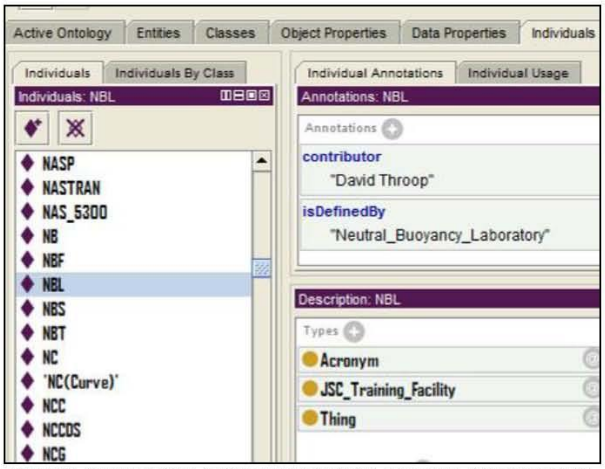


Figure 22: Snapshot of the Individuals Tab for 'NBL', a member of the '**Acronym**' and '**JSC_Training_Facility**' class

5.3 Acronym Verifier Plug-In

The Acronym Verifier plug-in verifies that each acronym's definition (the isDefinedBy annotation) is not only an annotation, but also exists as a member of the ontology; the plug-in then verifies that each acronym and its corresponding definition are members of the same class. Acronyms that fail either verification are exported to XLS files to be updated and then imported back to the Ontology.

1. Select 'Acronym Verifier' from the File drop down menu in the Protégé toolbar.

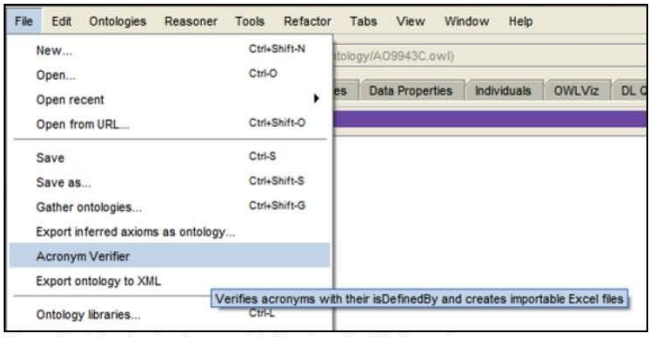


Figure 23: Selecting the Acronym Verifier from the File Drop down menu

2. Enter the file name and file location to export the data to.

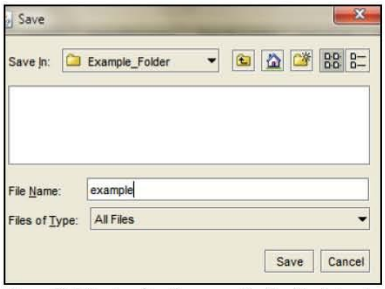


Figure 24: Naming the Acronym Verifier file to begin verification process

3. Click Save to start the verifications and export process.

   Upon completion, two XLS files will be created and stored in the designated file location from step 2, and a message will appear notifying the user of how many acronym definitions have been exported to each file.

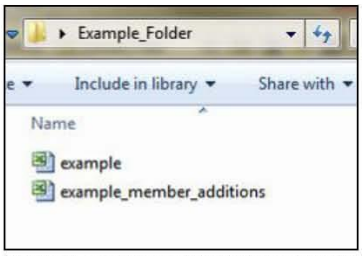
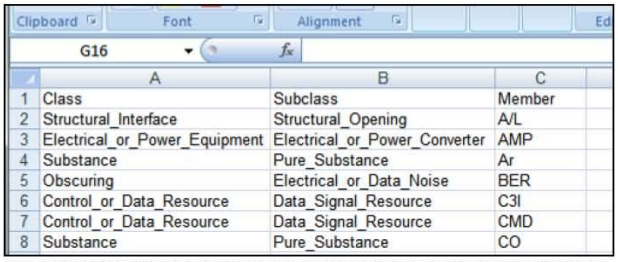


Figure 25: Acronym Verifier generates 2 XLS files

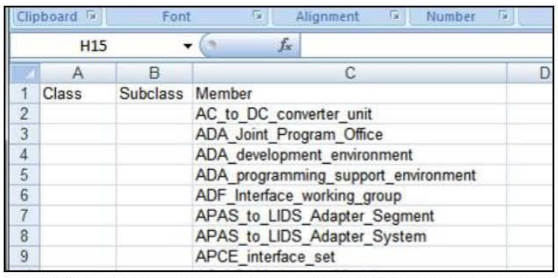
- XLS file #1: list all acronym members missing from their corresponding acronym definition's class, along with the name of the class and subclass that the corresponding acronym definition is a member of. File naming convention: XLS file name specified in step 2.

- XLS file #2: contains all the acronym definitions that are not yet members, along with space for the user to enter the class and subclass. Naming convention for this file is: XLS file name from step 2 with "_member_additions" appended to the end.



	A	B	C
1	Class	Subclass	Member
2	Structural_Interface	Structural_Opening	A/L
3	Electrical_or_Power_Equipment	Electrical_or_Power_Converter	AMP
4	Substance	Pure_Substance	Ar
5	Obscuring	Electrical_or_Data_Noise	BER
6	Control_or_Data_Resource	Data_Signal_Resource	C3I
7	Control_or_Data_Resource	Data_Signal_Resource	CMD
8	Substance	Pure_Substance	CO

Figure 26: XLS file#1: List of all acronym members that were missing classifications.



	A	B	C	D
1	Class	Subclass	Member	
2			AC_to_DC_converter_unit	
3			ADA_Joint_Program_Office	
4			ADA_development_environment	
5			ADA_programming_support_environment	
6			ADF_Interface_working_group	
7			APAS_to_LIDS_Adapter_Segment	
8			APAS_to_LIDS_Adapter_System	
9			APCE_interface_set	

Figure 27: XLS file#2: List of isDefinedBy definitions that need to become members

4. Open XLS file#1 first (file with the name given in Step 2). A list of each exported acronym along with its destination Class and Subclass is listed. Verify and Save any modifications.

5. Navigate to the Excel 2 OWL tab in Protégé.

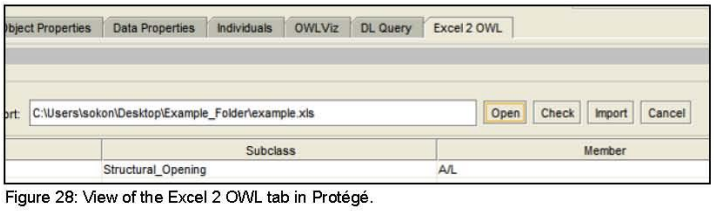


Figure 28: View of the Excel 2 OWL tab in Protégé.

6. Click Open and browse for the XLS file.
7. If modification to the file were made in Step 4, click Check.
   - Red cell - invalid entry. Check spelling in XLS spread sheet, make any necessary corrections, and repeat steps 4-7.
8. Click Import. Upon selecting import the acronym will be added to its designated class.
9. Open the XLS file#2 with the "_member_additions" appended file name. The user must identify and enter the class and subclass for each acronym definition member listed. Click Save.
10. Navigate to the Excel 2 OWL tab in Protégé. Click Open, Check, and then Import.

Note: At this point the steps are the same for when importing new data from and excel sheet, as described in sections 5.1 and 5.2.

6.0 Add a New Class

This section describes the steps of the third task in maintaining the Aerospace Ontology: Adding a new class to the Ontology.

6.1 Example 3: Determining the Need for a New Class

This example starts off where Example 1, in Section 4.0, ends. In Example 1, '*oblong*' was defined as a shape (noun) and also as an adjective to describe a particular geometric shape with certain properties, i.e. something that is longer than it is wide, etc. In addition we discovered a couple extra defining terms within the definition of '*oblong*': 'parallelogram' and 'equilateral'. Both parallelogram' and 'equilateral' are terms that can be used to describe a geometric figure… parallelogram describes a 4-sided geometric figure; equilateral describes a geometric figure in which all sides are of equal length. All three terms ('oblong', 'parallelogram', and 'equilateral') are used as shape descriptions, and hence qualify a new classification besides the class '**Shape**'.

Next, determine whether these terms should be added to an existing ontology class or if a new class will be added to the ontology. In Example 1 we explored the members of '**Shape_Property**' class, and determined that 'oblong' is a bit more specific than the other members. Next we can explore the subclasses of the '**Shape_Property**' class for potential fits. These two subclasses, '**Curve_Property**' and '**Propellant_Shape_Property**', do not appear to fit for our 3 new terms. We can conclude that a new class is required to account for our 3 terms.

Finally, determine a name for the new class. The three terms are descriptions of geometric figures, so we will name the new class: '**Geometry_Property**'.

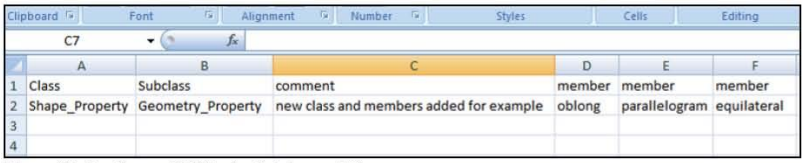
6.2 Example 4: Adding a Class to the Ontology

This example will demonstrate adding a new class (and subsequent members of the class) through use of the Excel 2 OWL tab (importing additions from an excel spreadsheet). To add classes to the Ontology manually through protégé see section 13.0: Manual Operations in Protégé.

Steps to Add an a New Class

For this example we will add a new class, entitled: '**Geometry_Property**', and designate '*parallelogram*', '*equilateral*', and '*oblong*' as members of the '**Geometry_Property**' class.

1. Start a new XLS file.



	A	B	C	D	E	F
1	Class	Subclass	comment	member	member	member
2	Shape_Property	Geometry_Property	new class and members added for example	oblong	parallelogram	equilateral
3						
4						

Figure 29: Creating an XLS file for Ontology additions

2. Enter the column headers in row 1. For this example we will be adding three members, so name the columns: Class, Subclass, member, member, and member.

   Row 1 will always contain the column headers (i.e. Class, Subclass, Members, etc.); no order is necessary for column headers (see Figure 29, row 1).

   Note: Reference section 3.1, Excel 2 OWL Tab for XLS file tips

3. Select the class in which you would like to create a subclass. For this example we will select '**Shape_Property**'. To locate '**Shape_Property**', you can either browse the Class hierarchy tree or use the search engine in Protégé.

4. Enter '**Shape_Property**' below column header Class.

5. Enter '**Geometry_Property**' below column header Subclass.

6. Enter '*parallelogram*', '*equilateral*', and '*oblong*' below the three columns entitled member.

7. Add annotations, such as comments, contributor, etc., to the XLS file. For a list of annotation headers supported by the Excel 2 Owl plug-in, see Section 3.1.

8. Save the XLS file (does not support .xlsx).

9. Next navigate to the Excel 2 OWL tab in Protégé.

10. Click Open and browse for the XLS file.

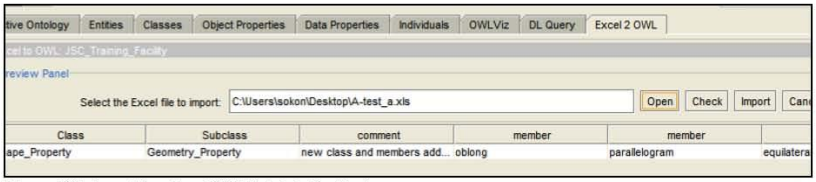


Figure 30: View of the Excel 2 OWL tab in Protégé.

11. Click Check to verify class and new member existence; check will only verify classes, subclasses, and members.

    - Green cell – a class or subclass with this name exists in the Ontology.
    - Red cell – a class or subclass with this name does not exist in the Ontology. Unless the user is creating a new ontology class, both columns should be green. For this example, '**Geometry_Property**' is a new class addition, and hence will turn red.
    - Blue cell – a member is new to the entire ontology. If a member is new to a class and not to the ontology, the cell will remain white.

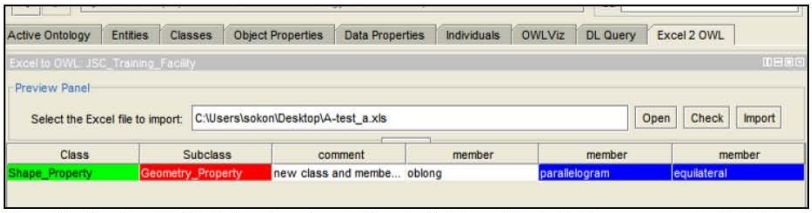


Figure 31: Checking XLS file before importing additions to Ontology. Blue indicates a new member to Ontology.

12. Click Import.

    If the import was a success the user will receive a message, with a list of all new classes and members added to the ontology (as a whole).

Figure 32: After importing additions, a message appears notifying user of new members and classes.

13. To verify new additions search for the class: '**Geometry_Property**' in the search field.

    Select '**Geometry_Property**' from the drop down and navigate to the Entities tab view, '**Geometry_Property**' will show up as a class in the Ontology on left, '**Shape_Property**' will show up in the 'Description:' frame on right below the 'Superclasses' header, the new members will be below the 'Members' header, and in the top right frame below the 'Annotation' header, will be the new comment (see Figure 33).

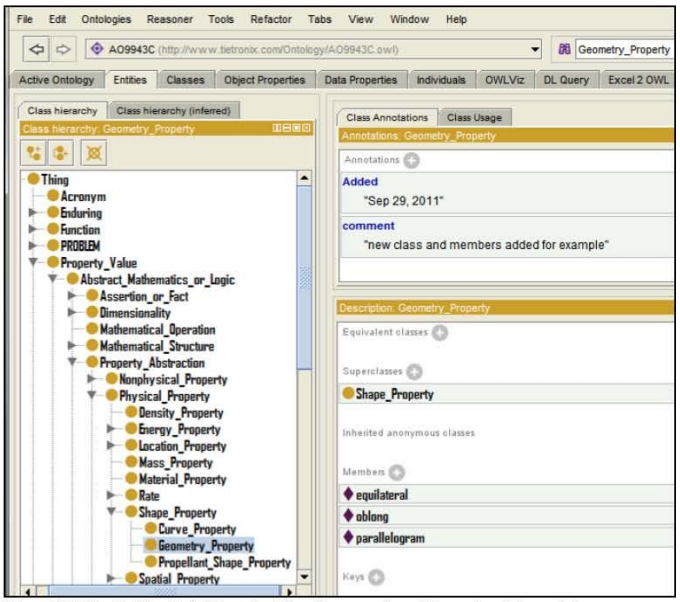


Figure 33: The Entities Tab View depicting the new class '**Geometry_Property**'

7.0 Rearrange the Class Hierarchy

The fourth task in maintaining the Ontology is to rearrange the Class Hierarchy. After adding a new class and/or Individual, it may be necessary to rearrange the class hierarchy to represent a more logical flow to the Ontology. The user can move a class in the hierarchy up a level, down a level, or combine as needed. This will be demonstrated with an example using the newly added class: '**Geometry_Property**' (See Section 6.0 for reference).

Some questions to consider for this task are:

- Is this the best location for the new class?
- Does the class belong above or below a particular class?

The user may start tackling these questions by simply looking for some definitions of "Geometry" on the web or other sources:

> ***Geometry*** *is a part of mathematics concerned with questions of size, shape, relative position of figures, and the properties of space; Initially a body of practical knowledge concerning lengths, areas, and volume; etc....*

With the above definition in mind we may decide to pursue a rearrangement of the class hierarchy. It appears that all the classes below '**Spatial_Property**' can also be considered a '**Geometry_Property**' (Area, Capacity, Linear, and Volume Properties are all aspects of Geometry). One might argue that Shape Properties are also Geometry Properties. *(Note: the ontology can become whatever the user deems it to be, and these changes are strictly for purposes of this example)*

Therefore, we can conclude that '**Shape_Property**' and '**Spatial_Property**' are both suitable subclasses of the '**Geometry_Property**'class.

7.1 Example 5: Rearranging the Class Hierarchy

The following steps demonstrate how to rearrange the class hierarchy in the Ontology. In this example, the goals are to make '**Geometry_Property**' a subclass of '**Physical_Property**' and a superclass of '**Shape_Property**' and '**Spatial_Property**'

1. Select '**Geometry_Property**' from the Class hierarchy in the Classes Tab view.
2. Navigate to the 'Description: Geometry_Property' Frame on right and select the 'Add' icon (+) to the right of 'Superclasses'.
3. Select '**Physical_Property**' from the Class hierarchy tab view. Click OK.
4. In the same section, under the 'Description' Frame select the (x) icon to the far right of '**Shape_Property**' to delete it from the list of 'Superclasses'.
5. Select '**Shape_Property**' from the Class hierarchy in the Classes Tab view and add '**Geometry_Property**' to its list of Superclasses (as done in Step 1). Select and delete '**Physical_Property**' (as done in Step 2).
6. Select '**Spatial_Property**' from the Class hierarchy in the Classes Tab view and add '**Geometry_Property**' to its list of Superclasses (as done in Step 1). Select and delete '**Physical_Property**' (as done in Step 2).
7. Review results. '**Geometry_Property**' should now reside as a subclass of '**Physical_Property**' and a superclass of the '**Shape_Property**' and '**Spatial_Property**' classes (see Figure 34).

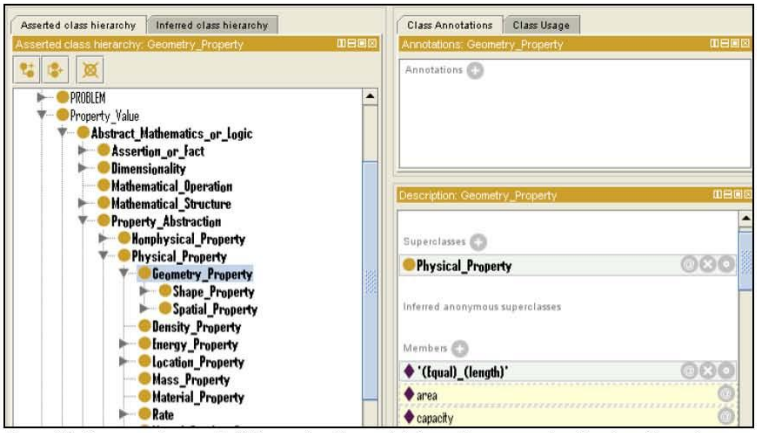


Figure 34: The new placement of '**Geometry_Property**' class after rearranging the class hierarchy

8.0 Validate and Verify Modifications to the Class Hierarchy

Now that additions to the ontology have been made, some of which may have potentially changed the hierarchical order, it is important as the final step in maintaining the Ontology, to check that the Ontology is correct.

The Reasoner (also known as the classifier) can help check for any inconsistencies in the class structure. The class hierarchy that is automatically computed by the Reasoner is called the ***Inferred hierarchy***.

8.1 Using the Reasoner

1. To classify the ontology, select 'FaCT++' from the Reasoner drop down menu (see Figure 35).
2. Verify that the Ontology is classified by selecting the Classes Tab or Entities Tab and then the Inferred hierarchy tab that appears in the class hierarchy view. (Note: it might take a few seconds for the Inferred hierarchy to populate) If you see only the root class, '**Thing**', the Ontology may not be classified.
3. If any item is highlighted in red, it indicates that the Reasoner has found this class to be inconsistent. If any items appear in a blue color, it means that the class has been reclassified (i.e. its superclass has changed).
4. You can also select Classify from the Reasoner drop down menu to classify the Ontology, but ONLY after FaCT++ has been used at least once.

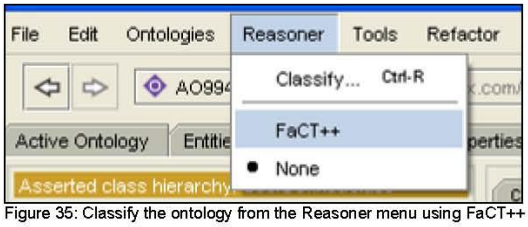


Figure 35: Classify the ontology from the Reasoner menu using FaCT++

9.0 Add a User-Defined Class

9.1 The UserDefinedClassifier Class

The '**UserDefinedClassifier**' class is the last class in the list directly under '**Thing**'. The '**UserDefinedClassifier**' class currently has only 4 members: '*electrical_thing*,' '*mechanical_thing*,' '*process_thing*,' and '*software_thing*.' The '**UserDefinedClassifier**' class was added to the Ontology for a specific reason.

Before the Ontology was created in Protégé, there were 4 separate versions of the Ontology, Electrical, Mechanical, Process, and Software Ontologies. The STAT tool was able to read and tag documents using terms from the 4 Ontologies. These 4 Ontologies all worked collectively for the Aerospace Ontology.

When the 4 ontologies were consolidated into one OWL Ontology in Protégé, it was decided that an additional class be made, the User Defined Classifier class. This class would contain Individuals that pertain to the description of the 4 categories of the 4 original Ontologies: '*electrical_thing*,' '*mechanical_thing*,' '*process_thing*,' and '*software_thing*'. Relationships between the Individuals of the User Defined Classifier class and the other terms in the Ontology would be identified so that AO users would know where the terms originated.

For example, the term '*hammer*' came from the '**Mechanical**' Ontology… When the term '*hammer*' is looked up in the AO, it is described as: '*hammer*' – has User Defined Classifier '*mechanical_thing*.' Another example: The term '*drive*' existed in both the '**Mechanical**' Ontology and the '**Software**' Ontology. In the description for '*drive*,' it states: has User Defined Classifier '*mechanical_thing*' **and** '*software_thing*.'

9.2 Add a New Term to the UserDefinedClassifier Class

The following steps are the same as adding a new term to the Ontology (section 5):

1. Create an XLS file to create the additions.
2. Add a column header: Class and member.

3. Enter '**UserDefineClassifier**' below the column Class and add the new classifier below the column member.
4. Save the XLS file (does not support .xlsx).
5. Navigate to the Excel 2 OWL tab in Protégé.
6. Select Open, to navigate and open the XLS file.
7. Select Check to verify additions.
8. Select Import to add addition.

9.3 The Object Property in Protégé

An Object Property in Protégé creates relationships between Individuals in the Ontology. For the AO ontology we have only one Object Property defined and that is the '**hasUserDefinedClassifier**' property. Currently the only relationship in the Aerospace Ontology is the relationship between the Individuals of the '**UserDefinedClassifier**' class and the Individuals of the remaining classes in the Ontology. Since this relationship between the Individuals is to a member of the UserDefinedClassifier class, it was most suitable to name this property '**has**UserDefinedClassifier'.

9.3.1 Adding an Object Property

Although '**hasUserDefinedClassifier**' is currently the only object property in the AO, in the future the Ontology may need modifications and object properties may need to be added. Figure 29 depicts the various buttons that can be used to add and delete object properties.

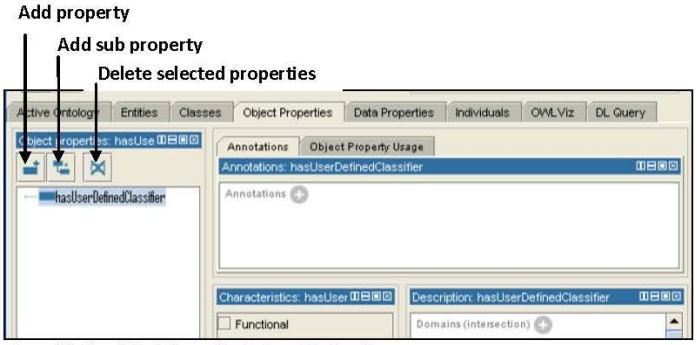


Figure 36: The Object Properties Tab and its functions

1. Select the Object Property Tab
2. Select 'Add Property'

3. Enter the name of the Object Property then click OK

9.3.2 The Recommended Naming Convention for the Object Property

The naming convention used for the object property in the Aerospace Ontology, '**has**UserDefinedClassifier' is the recommended naming convention. Property names start with a lower case letter, have no spaces, and have the remaining words capitalized. It is also recommended that properties are prefixed with the word 'has' or the word 'is' which makes its intent much easier to recognize and understand. There is no strict naming convention for properties, and this is simply recommendations and to better explain the way the AO object property is depicted.

9.3.3 Add an Object Property to a New Term

New terms can be given a Classifier description from the '**UserDefinedClassifier**' class if they belong to one or more of the four defined Classifier descriptions. The Excel 2 OWL tab assists users with entering classifiers to new terms.

After a file is imported into the Ontology via the Excel 2 OWL tab, a new file is generated following the naming convention of the imported file with "_classifier" appended to the end, and stored in the same location as the imported file.

1. Navigate to the auto-generated XLS sheet with the "_classifier" file name. For this example we will refer back to the file created for import in example 1.

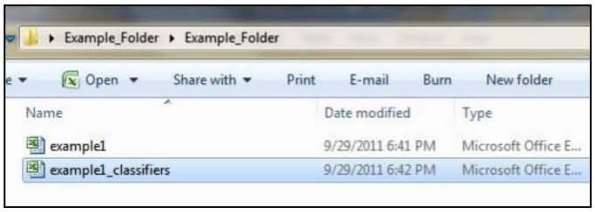


Figure 37: Excel 2 Owl function generates an XLS file for classifiers.

2. Open the new XLS file.

   The XLS will contain a list of newly added members and their potential classifiers. These classifiers are based on an internal query of all the members of a class (so if a class has members with multiple classifiers, a row for each member and classifier will be displayed), see Figure 38.

	A	B	C
1	Member	ObjectProperty	Member
2	oblong	hasUserDefinedClassifier	mechanical_thing
3	oblong	hasUserDefinedClassifier	software_thing
4			
5			

Figure 38: Format of auto-generated classifier XLS file.

3. Add/Delete/Modify the list of members and classifiers as necessary. Object properties can also be modified. When complete click Save.
4. Navigate to the Excel 2 OWL tab in Protégé.
5. Click Open and browse for the XLS classifiers file.
6. Click Check
   - Green cell – members and object properties exist in ontology.
   - Red cell – invalid; row containing red-cell will not be imported. Check spelling in XLS spread sheet, make any necessary corrections, and repeat steps 6 – 9.
7. Click Import. Upon selecting import the classifiers (and object properties) will be added to members in the Ontology.

To clear data in the Excel 2 OWL tab, click Cancel.

10.0 Add Annotations to the Ontology

Annotations can be added to the Ontology to add descriptions to classes, members, or properties, to explain additions and changes, assign contributor or dates, etc. Annotation can be used for documentation purposes. The user can add annotations to the Active Ontology, Entities, Classes, Individuals, and Property Tabs in Protégé. If adding annotations to classes, the Excel 2 OWL plug-in can support these additions, as explained in section 3.1. All other annotations must be added manually as follows:

1. Select the 'Add' icon (+) to the right of the 'Annotations' header located in the 'Annotations:' frame.
   - For Active Ontology Tab: 'Ontology annotations:' frame is located at top left (see Figure 39).
   - For Entities, Classes, Individuals, or Property Tabs: 'Annotations:' frame is located at top right.
2. Select the annotation type from the list on left and enter the value (i.e. comments, name, description, etc.) under the Constant tab on right (see Figure 39). Then click OK.

   The annotations will appear below the 'Annotations' frame.

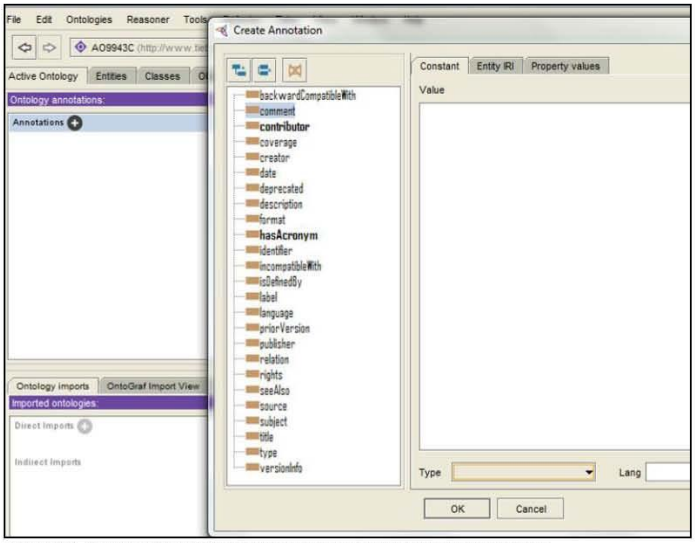


Figure 39: Selecting 'Add Annotation' to chose annotation types and values

11.0 Individuals with Complex Word Equations

Some Individuals in the Aerospace Ontology are more complicated than just a single word; these Individuals can be represented in complex appearing word-equations. We will look at a couple examples to help better understand the meanings of these complex word equations.

Note: these representations of word equations originated from the 4 original ontologies made in Microsoft Word. See Appendix B for more information.

11.1 Example 6: Understanding Complex Members

Let's begin with a simple example. One of the Individuals in the class '**Electrical_Imbalance**' is: *'(Excessive,_Excessive_Value)_(voltage,_current)'*

This should be read as follows:

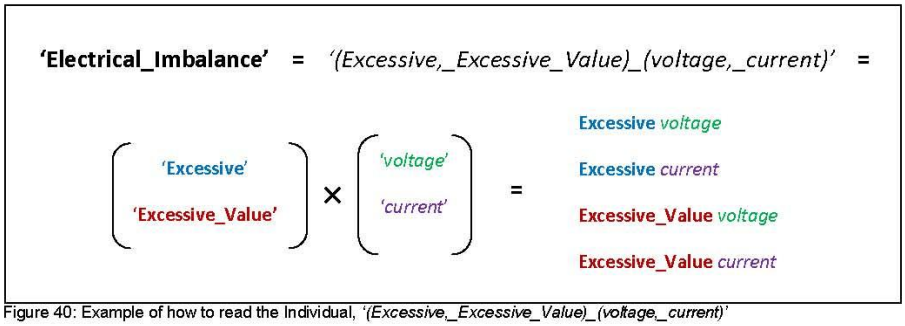


Figure 40: Example of how to read the Individual, *'(Excessive,_Excessive_Value)_(voltage,_current)'*

In the above example, the **bolded** words with a Capitalized letter represent classes from the Aerospace Ontology. The italicized lowercase words represent members from the Aerospace Ontology.

To further breakdown this complex member, we will look at one of the four results from Figure 40 above; let's use the term '**Excessive_Value** *voltage*'. Once again note that Excessive_Value in **bold** is a class and voltage in *italics* is an Individual. This term should be read as follows:

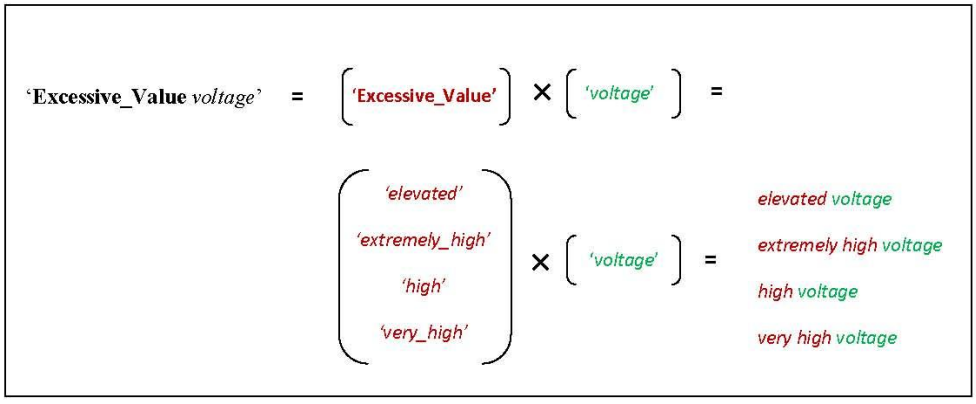


Figure 41: Example of how to read and breakdown the term, '*Excessive_Value voltage*'

In conclusion, the member, *'(Excessive,_Excessive_Value)_(voltage,_current)'*, allows for a consolidated equation to represent multiple terms. The results in Figure 40 produced 4 terms, of which we took one and expanded down to its lowest form, composed of members (Figure 41), the same can be accomplished with the remaining 3 results in Figure 40.

The STAT tool would use this complex member in the ontology to find documents containing terms similar to Electrical Imbalance, such as 'Excessive voltage', 'Excessive_Value current', 'high voltage', 'elevated voltage', 'extremely high voltage', 'very high voltage', etc.

Next let's look at a slightly more complex example, a complex member of the Class '**Measure**': *'(measure,_determine,_calculate,_get)_((Measurement),_[Physical_Property],_[Nonphysical_Property])'*.

In complex members, classes in (Parenthesis) refer to every member in that particular class, and classes in [Brackets] refer to all subclasses and the members of the class and its subclasses. This complex member should be read as follows:

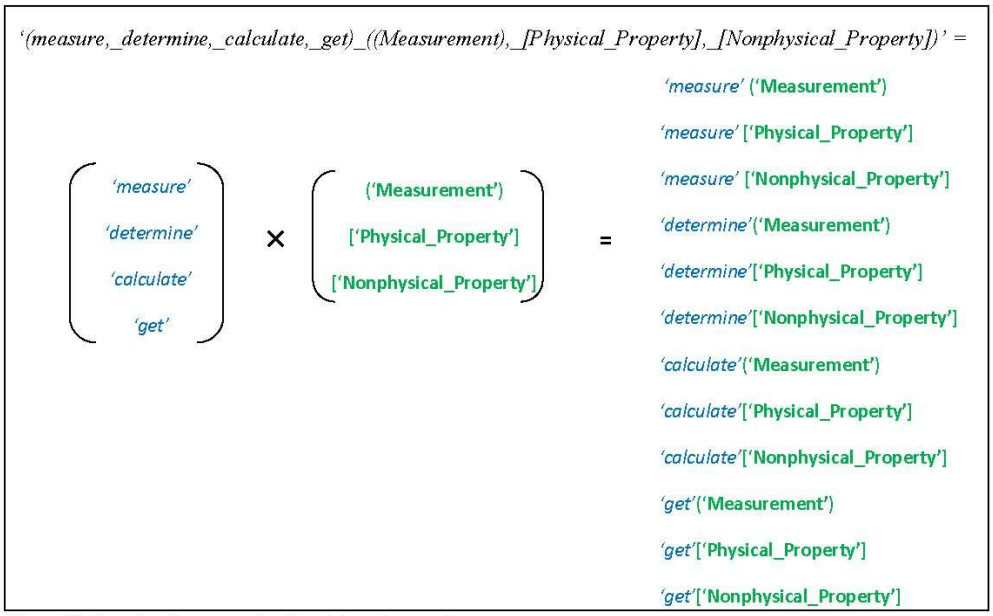


Figure 42: Example of how to read and breakdown the complex term.

The complex member in Figure 42 above results in 12 terms made up of combinations of individual members and classes from the Aerospace Ontology. The above 12 results can be further broken down as follows:

- The member *'measure'* along with every member of the class **'Measurement'**
- The member *'measure'* along with the subclasses and members of the class **'Physical_Property'** and its subclasses (can be repeated with **'Nonphysical_Property'**)
- The member *'determine'* along with every member of the class **'Measurement'**
- The member *'determine'* along with the subclasses and members of the class **'Physical_Property'** and its subclasses (can be repeated with **'Nonphysical_Property'**)
- The member *'get'* along with every member of the class **'Measurement'**
- The member *'get'* along with the subclasses and members of the class **'Physical_Property'** and its subclasses (can be repeated with **'Nonphysical_Property'**)

Complex members can represent numerous combinations of members and classes within the Ontology. Complex members can be added to the Ontology following the same steps described in Section 5.0, Example 2. When creating the XLS file with the new members refers to the XLS file creation tips noted in Section 3.1. Avoid the use of spaces and single quotes, use underscores where spaces are needed.

11.2 Tips to Remember When Reading a Complex Individual

Lowercase terms represent Individuals

Terms that start with an Uppercase letter represent Classes

Classes in (Parenthesis) = every individual from the Class; i.e.: (Measurement)

Classes in [Brackets] = All the classes and subclasses, along with the Individuals of its class and subclasses; i.e.: [Physical Property]

12.0 Helpful Tools in Protégé

12.1 OWL Viz Plug-In

The OWL Viz Tab allows the user to visualize the Asserted and Inferred Class hierarchies using a model diagram. *To use OWL Viz, you will need to download the OwlViz plug-in available from the CO-ODE website or it can be installed when Protégé 4 is installed (see the Helpful Links Section). Note: Graphviz must also be installed.*

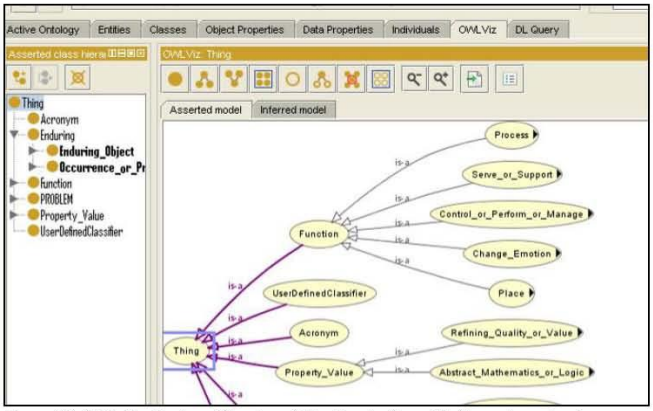


Figure 43: OWL Viz display of the class 'Thing' and a few of its "lower branches"

Figure 43 depicts the Asserted model for the class, '**Enduring**'. The object selected in the tree will be highlighted on the left and represented with a square in the model on the right. The model depicts the 2 subclasses of Enduring and each of their subclasses. A black arrow in the corner of a class means there are additional subclasses (or additional superclasses, depending on the direction of the arrow).

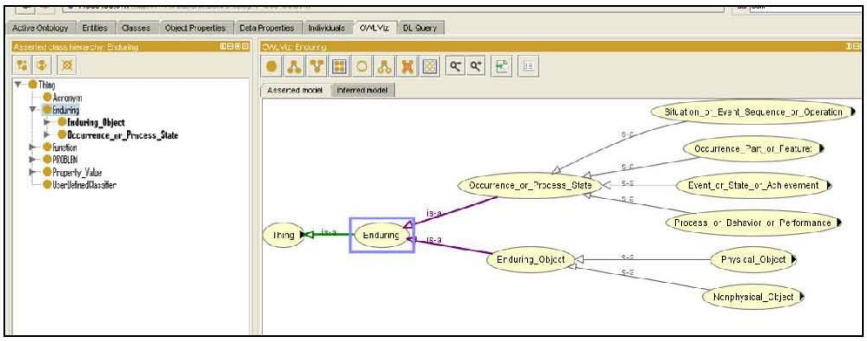


Figure 44: OWL Viz display of the class '**Enduring**', highlighted and in a square above

OWL Viz provides the user the ability to hide and show classes, select the radius of the model (how many branches it will expand), as well as the layout (horizontal or vertical), see Figure 45 and 46:

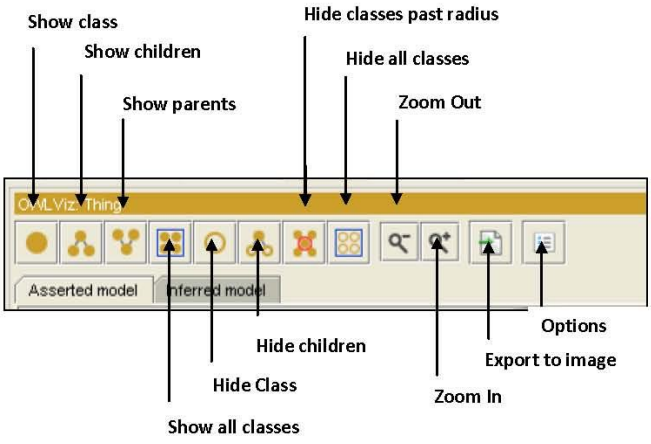


Figure 45: The OWL Viz Pane

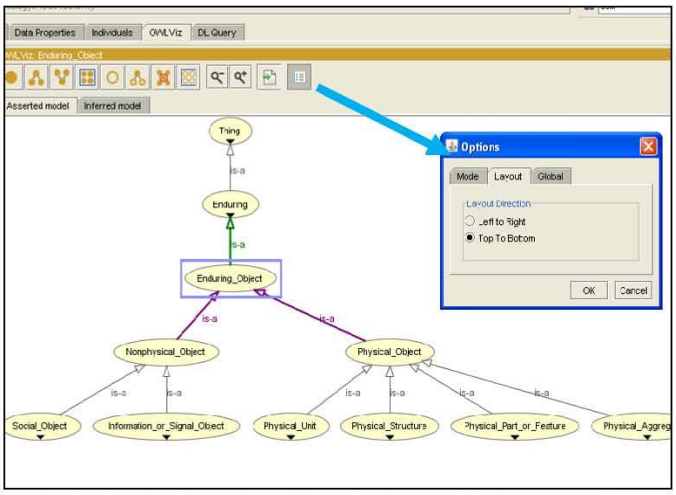


Figure 46: A depiction of the Options tool in OWL Viz and the Top to Bottom display

12.2 DL Query Plug-In

The DL Query allows users to view information about selected classes. The user can select options that include viewing the class's appropriate subclasses, superclasses, Individuals, etc.

Using DL Query:

1. Ensure that the DL Query Tab is selected (Figure 40)
2. We first must validate that the Ontology is classified prior to executing a query. To do this we must use the Reasoner (*see Section 8.1. Using the Reasoner*).
3. Place the object in the 'Query (class expression)' section that you wish to query (i.e. a class name). Select from the very right column whether you want to find the object's superclasses, subclasses, Individuals, etc…, or any combination of these choices.
4. If the object is written correctly, the 'Execute' button will become available. Hit Execute and the results from the Query will be made available under 'Query results.'

Note: *If the following message appears: "Reasoner out of sync", perform Step 2 again (See Figure 47)*

Figure 47: "Reasoner out of Sync" message

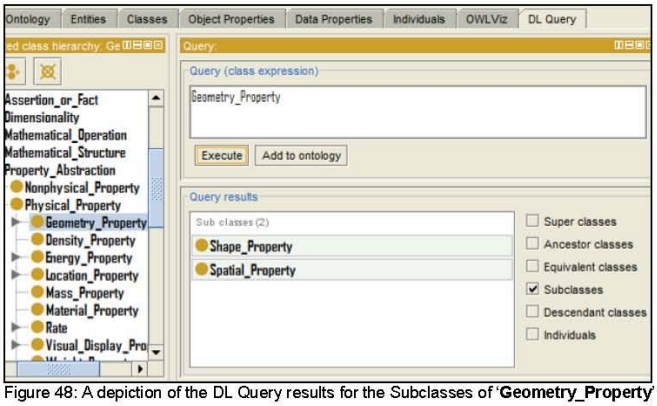


Figure 48: A depiction of the DL Query results for the Subclasses of '**Geometry_Property**'

12.3 Export to XML Plug-In

The Export to XML plug-in allows the user to export data contained in Protégé to an .xml file to be used in other applications as required.

1. Select 'Export to XML' from the File drop down menu in the Protégé toolbar. (See Figure 49)
2. Enter the following file name: "vers 1.07 Aerospace Ontology.xml". Specify a location to export the data to.
3. Click Save to begin export.

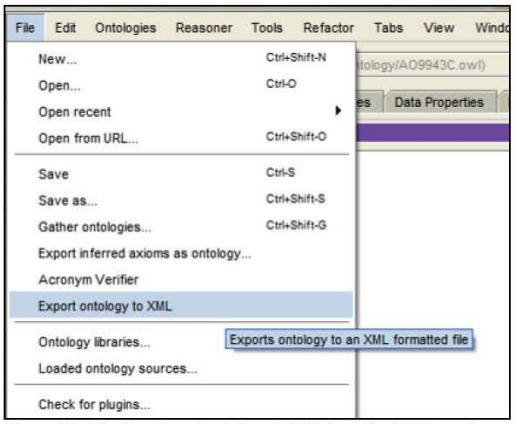


Figure 49: Selecting Export ontology to XML from the File Drop down menu

4. A pop-up message will appear asking the user to input a tag name for each of the 6 main Ontology classes (see Figure 50). Input the following tag names when prompted by the Export ontology to XML plug-in:

   Type ACRONYM when asked for a tag name for Acronym

   Type NOUN when asked for a tag name for Enduring

   Type VERB when asked for a tag name for Function

   Type FAILURE when asked for a tag name for PROBLEM

   Type PROPERTIES when asked for a tag name for Property_Value

   Type USERDEFINED when asked for a tag name for UserDefinedClassifier

Figure 50: Inputting a tag name for export to XML

5. A message will appear upon completion, notifying the user that the export is complete. Click OK to continue.

13.0 Manual Operations in Protégé

The recommended method for updating the Aerospace Ontology is to use an XLS spreadsheet along with the Excel 2 Owl tab, as mentioned and demonstrated in the previous sections. However, member, class, and object property additions can also be accomplished via manual operations within the Protégé tool. The following Examples will demonstrate manual operations in Protégé.

13.1 Example 2 with Manual Operations: Adding a New Term

The following steps demonstrate example 2, entering a new term into the Ontology. This example will demonstrate adding members through manual use of the Protégé tool.

1. Switch to the 'Classes Tab' shown in Figure 51 (For Protégé 4.1.0 the user can go directly to the 'Individuals Tab', the Class hierarchy column will be available on left):

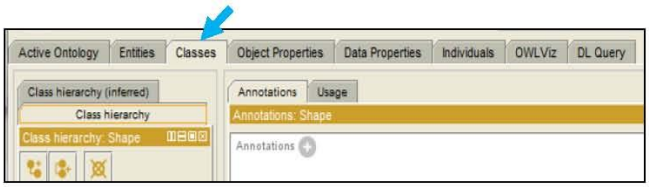


Figure 51: The Classes Tab

2. Select the class that the new term will become a member of.

   For this example, select the class '**Shape**' from the Class hierarchy Tab on left. The class is selected when it is highlighted in light blue. (*Note: The search engine can also be used to locate the class*)

3. (For Protégé 4.1.0 the user can skip step 3)

Navigate to the 'Description: Shape' frame located at the bottom half of the Classes Tab view.
Select the 'Add' icon (+) to the right of the 'Members' header.
A pop-up of a list of Individuals (similar to that under the Individuals Tab) will appear:

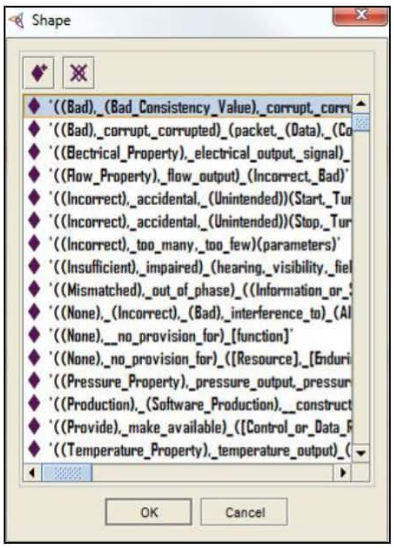


Figure 52: The Pop-up that appear upon selecting the Add icon (+) next to 'Members'

4. Press the 'Add individual' button shown in Figure 53

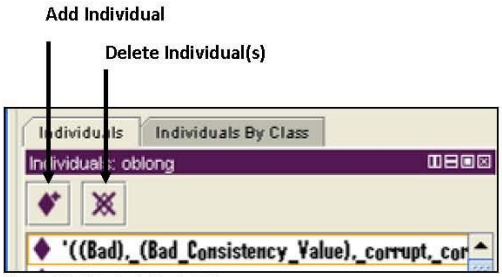


Figure 53: The Individuals Tab

5. Name the new Individual 'oblong', see Figure 54.

   Then click OK.

Figure 54: Naming the individual

6. Click OK again (For Protégé 4.1.0 the user will click OK once), and '*oblong*' will then appear in the list of Members for the class '**Shape**' (See Figure 55):

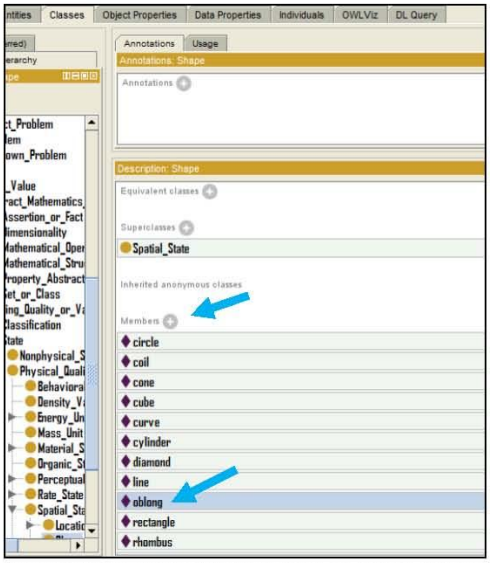


Figure 55: Depicting the list of Members with the addition of 'oblong'

7. To add another class 'Type' to the Individual, switch to the 'Individuals Tab' (Figure 56):

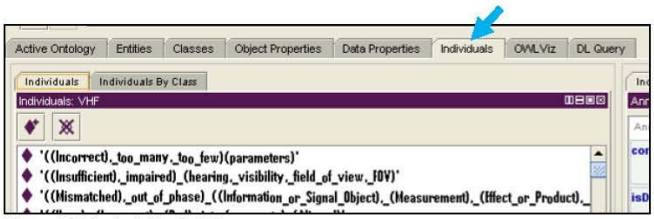


Figure 56: The Individuals Tab

Select '*oblong*' from the list of Individuals

Navigate to the 'Description: oblong' frame in the center of the window. Select the 'Add' icon (+) next to the 'Types' header (See Figure 58).

Choose '**Thing**' from the Asserted class hierarchy Tab, then click OK (See Figure 57):

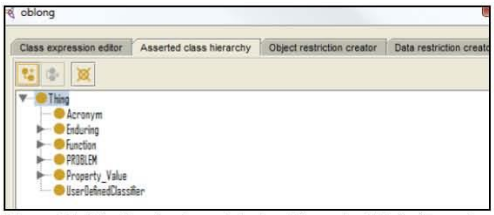


Figure 57: Selecting the Asserted class hierarchy Tab for 'Types'

This makes the Individual '*oblong*' a member of the class '**Thing**' and '**Shape**'. '**Thing**' and '**Shape**' now appear below 'Types':

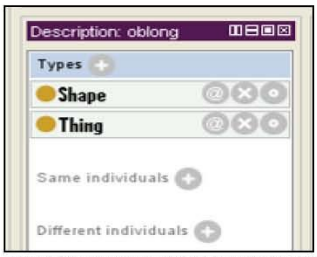


Figure 57: The Description frame for 'oblong'

*****Additional Practice**********

For additional practice, add '*diamond*' and '*rhombus*' as Individuals to the class '**Shape**'.

13.2 Adding an Acronym Member with Manual Operations

The following steps will demonstrate how adding an acronym to the Aerospace Ontology is accomplished via manual operations in Protégé:

1. Switch to the Classes Tab
2. Select the class, '**Acronym**' (when the class is highlighted in light blue it has been selected)
3. Click the Add icon (+) to the right of the 'Members' header below the 'Description' frame and a pop-up with a list of Individuals will appear.
4. Press the 'Add individual' button (shown in Figure 53)
5. Type in the acronym to name the Individual (i.e. NBL). Then click OK, and the acronym should appear in the long column of Individuals.

   Click OK again, and the Individual shall appear as a member of the class.
6. If the acronym is to be a member of an additional class, select the additional class from the Asserted class hierarchy and perform steps 3-5.

   This will make the Individual a member of the class '**Acronym**' and any additional class.
7. Next, switch to the Individual Tab. Search for the acronym from the list of Individuals on the left side. Select the acronym.

   Select the 'Add' icon (+) to the right of the 'Annotations' header, located below the Individual Annotations Tab. You will see Figure 59

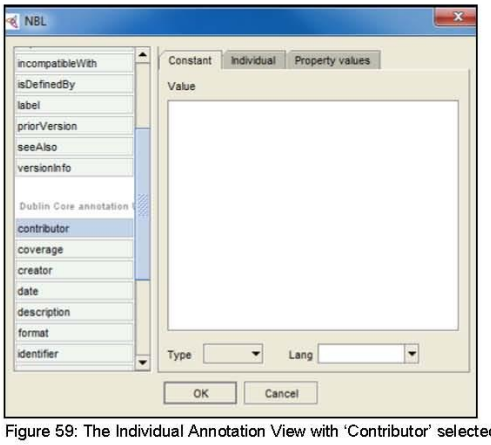


Figure 59: The Individual Annotation View with 'Contributor' selected on the left side

Choose 'contributor' form the list on left and enter the name of the person who is making this addition under the Constant tab on right. Then click OK.

You should see this input entered below the 'Annotations' frame (Figure 61).

8. Then select the Add icon (+) to the right of the 'Annotations' header again.

   This time choose 'isDefinedBy' from the list on left and enter the full name of the acronym below the Constant tab on right. (Figure 60)

   Then click OK.

   (If the acronym has multiple meanings, repeat step)

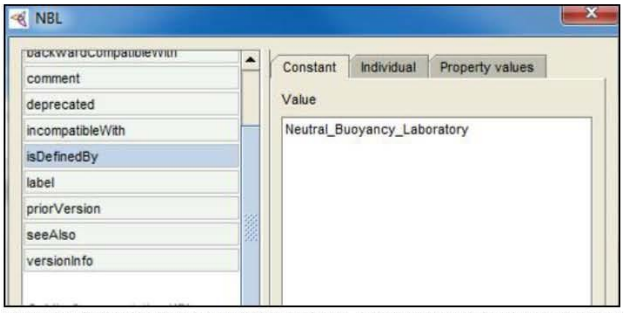


Figure 60: Individual Annotations View for '*NBL*', with 'isDefindeBy' selected on left column and the definition of the Acronym under the Constant Tab on right side.

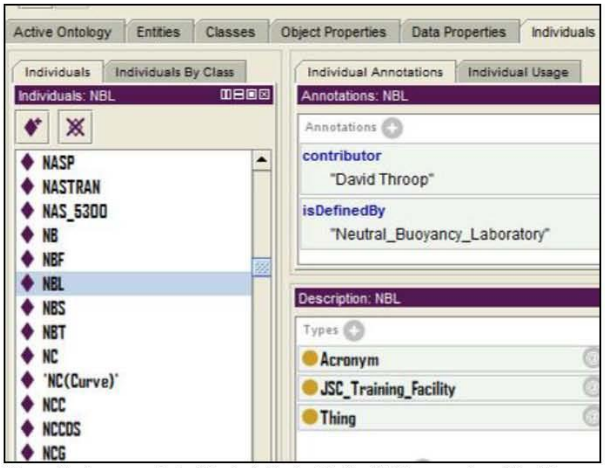


Figure 61: A screen shot of the Individuals Tab for 'NBL', a member of the '**Acronym**' class and the '**JSC_Training_Facility**' class

13.3 Example 4 with Manual Operations: Adding a Class

The following steps demonstrate example 4, entering a new class into the Ontology. This example will demonstrate adding classes through manual use of the Protégé tool.

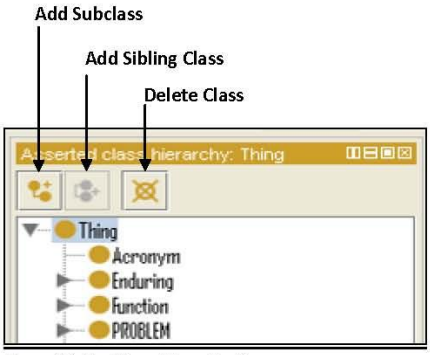


Figure 62: The Class Hierarchy Pane

1. Ensure that the 'Classes Tab' is selected.

2. Select the class in which you would like to create a subclass. For this example locate and select '**Shape_Property**' (will be highlighted in light blue). To locate '**Shape_Property**', you can either browse the Asserted class hierarchy tree or use the search engine.

3. Press the 'Add subclass' button shown in Figure 62. This button creates a new class as a subclass.

4. A dialog will appear to enter a class name, for this example we will enter '**Geometry_Property**'. If a valid name is entered, the OK button will become available. See Figure 63:

Figure63: Naming a new OWL Class

5. Click OK and '**Geometry_Property**' will show up as a class in the Ontology on left and '**Shape_Property**' will show up in the 'Description:' frame on right below the 'Superclasses' header (see Figure 64):

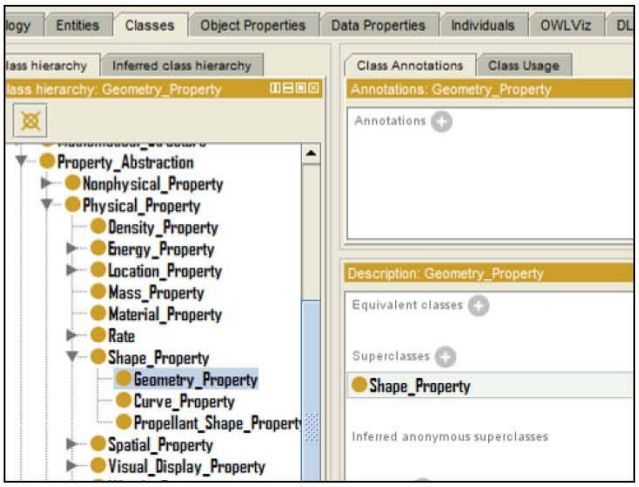


Figure 64: The Class Hierarchy view with the new class addition: '**Geometry_Property**'

*Tip: Another method to enter a new class is to select one of the subclasses of '**Shape_Property**', either '**Curve_Property**' or '**Propellant_Shape_Property**', and press the 'Add Sibling Button' (see Figure 62). Type in "Geometry_Property*" and click OK.

6. Enter '*parallelogram*', '*equilateral*', and '*oblong*' to the Ontology as Individuals and members of the '**Geometry_Property**' class (*Follow instruction in Section 13.1: Adding Members to a Class).*

7. View results. Select either the Entities Tab or Classes Tab view, and select the new '**Geometry_Property**' class from the class hierarchy, see Figure 65.

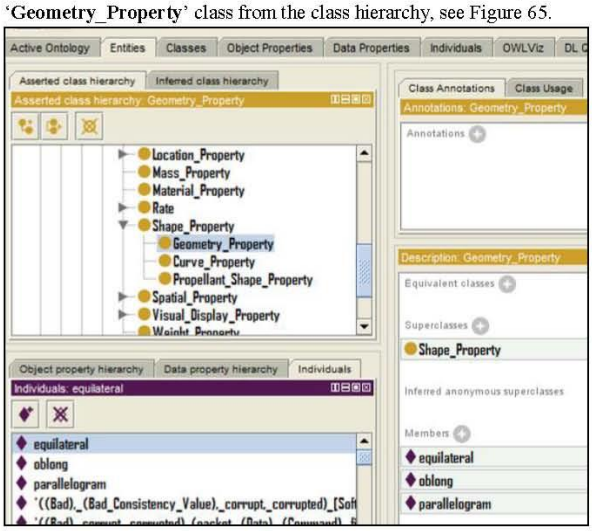


Figure 65: The Entities Tab View depicting the new class '**Geometry_Property**'

13.4 Adding an Object Property to a New Term with Manual Operations

1. Ensure that the 'Individuals Tab' is selected

2. Select the Individual from the long list on right that you wish to add a category description to. (i.e. the new term added in Section 5: '*oblong*')

3. Select the 'Add' icon next to the 'Object property assertions' header from the Property assertions frame located on the bottom right of the Individuals tab (See Figure 66).

4. Select the object property '**hasUserDefinedClassifier**' from the left column

5. Then select one of the members from the '**UserDefinedClassifier**' class from the long list of Individual on right. Select either '*electrical_thing*,' '*mechanical_thing*,' '*process_thing*,' or '*software_thing*' from the list of Individuals from the right column.

6. Click OK and the object property will be added under the Object property assertions as shown in Figure 66:

Figure 66: Adding an Object Property relating the Individual '*oblong*' to the User Defined Classifier '*mechanical_thing*'

Appendix A. The Active Ontology Tab/Viewing Metrics

The Active Ontology Tab provides information about the Ontology (i.e. metrics and annotations). For Protégé 4.1.0, the metrics, along with other view options can be found under the drop down menu: View | Ontology views | Ontology metrics (see Figure 67):

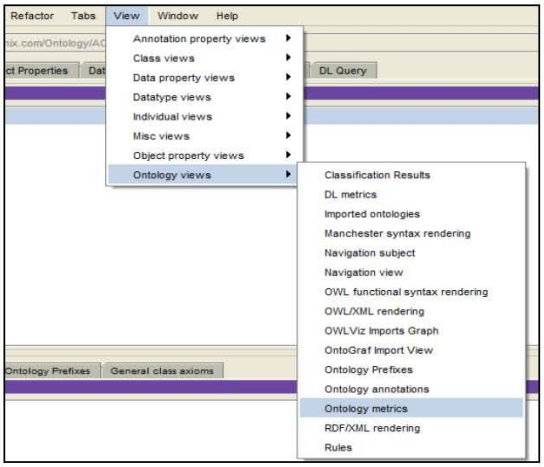


Figure 67: Navigating to the Ontology metrics in Protégé 4.0.1

Appendix B. Interpreting Individuals from the Microsoft Word Ontology Document

In Word document:	In Aerospace Ontology:
{ } < > Space (ex: solid model)	( ) [ ] Underscore (ex: solid_model)

Table 2: Definitions of Aerospace Ontology Semantics as compared to the original word document

Appendix C. Description of the Four Main Classes

- The **Enduring hierarchy** holds, roughly, all the nouns in the ontology.
    - This hierarchy provides more detailed classes and mapping words for objects, descriptions, occurrences and features/parts in the Ontology.
    - Entities include types of equipment, substances, regions, and interfaces.
        - The Entity hierarchy plays the roles of participants in descriptions: Performer/Agent/Actor, Instrument, Resource, Product or Patient/Operand.
        - These entities can play the roles of patient/operands, agents, instruments and resources.
- The **Function hierarchy** (Capability or Action Verb) holds the verbs. Because there are meta-functions (functions that operate on other functions, like prevent depressurizing), Functional Entities appears in the Enduring hierarchy; the functions are expressed as verbs, as actions that can be viewed as part of specifications or as part of occurrences.
    - The Function/Action Hierarchy classifies functions and actions for processing, placing, serving, energizing and controlling/performing.
        - The organization and contents of the Control/Manage/Perform class are influenced by work on software goals and on distinctions in organizational and cognitive psychology.
- The **PROBLEM class hierarchy** classifies encoded qualities of entities or functions that represent effectiveness or safety problems. The qualities are given as adjectives (using the Properties and Values Class Hierarchy) for entities or functions, but they can also have noun forms (e.g., anomalous or anomaly; noisy or too much noise).

- This hierarchy distinguishes types of damage, hazards, impairments, failures and deficiencies. It can be used to identify and categorize information about risks, symptoms and causes.

- The **Property_Value classes** hold the adjectives and adverbs.

Appendix D. The Data Properties Tab

Figure 67 depicts the Data Properties View. Data properties are not used in the Aerospace Ontology. Data Properties describe relationships between terms and data values. In the Aerospace Ontology none of the terms are defined to have a specific value so it was not necessary to utilize this particular function at this time. An example of a data property would be depicted as: '**hasSomeTypeOfValue**', where an example of an object property (*Section 9*) is: '**hasValue**'.

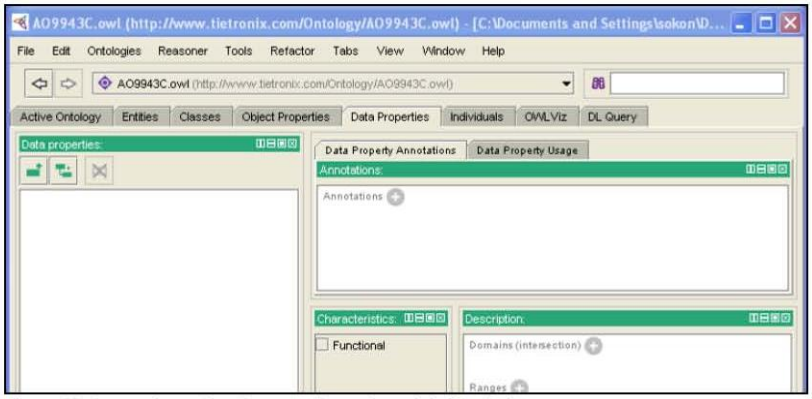


Figure 68: A snapshot of the Datatype Properties tab in Protégé

Appendix E. Helpful Links

http://protege.stanford.edu

http://www.co-ode.org

Appendix E. Tutorial: Analyzing Problem Reports with Flamenco+

Tutorial: Analyzing Problem Reports with Flamenco+

Example FAA Data Base

- 2007 Incident Reports

Incident Narrative: The helicopter was destroyed by **impact** forces and a post **crash fire** while attempting an auto-rotation after a mechanical **failure** . The pilot stated that he was practicing takeoffs , hovering , and quick stops above the runway . After takeoff , at about 50 feet and 50mph, he lowered the collective to initiate a quick **stop** . At this point the engine RPM revved up **out of control** . He pulled up on the collective to re-engage the **drive** system , but the system would **not** engage . He entered into an auto-rotation , and as he neared the ground the helicopter began to slide sideways , folding the skids under the helicopter . The helicopter then rolled on its side , and the occupants climbed out prior to the post **crash fire** . Subsequent examination of the helicopter \'s **drive** system revealed that the **drive** shafts , pulleys , and **drive** belts were intact . The only device within the helicopter **drive** system that could **not** be determined to be in working **condition** was the sprag clutch which transmits engine power to the rotor **drive** system . The sprag clutch was not examined during the course of the investigation .

Incident Cause: The **failure** of the sprag clutch which resulted in the **disengagement** of the drive unit and the pilot \'s misjudged landing during the autorotation . A factor was the low altitude at which the failure occurred .

Equipment_Involved: The **failure** of the sprag clutch which resulted in the disengagement of the drive unit and the pilot \'s misjudged landing during the autorotation . A factor was the low altitude at which the **failure** occurred .

2

Goals

- Discover high value problem groups
  - Frequency, consequences, opportunities for improvement
- Refine problem groups – common corrective action
  - Narrow: Find problems with a common combination of features. Narrow search by adding more features.
  - Broaden: Find additional problems that share some but not all features of a known example problem.
- Outputs graphs and spreadsheets

3

Faceted Browsing with Flamenco+

- Web-oriented browsing and search
- Recommended browser: Firefox
- Easy filtering and combining constraints
- Terminology
  - Item: A detailed description of some entity such as a problem report or incident report.
  - Instance: The set of items comprising a Flamenco+ database.
  - Facet: An item property whose values may be shared by multiple items, thus allowing items to be grouped together on in a subset of the Flamenco Instance's items.
  - Attribute: An item property whose values are not generally shared by multiple items and thus cannot be used to group items. Attributes contain the information unique to an item.
  - Facet Constraint: The value for a selected facet that all items in the selected set must share.

4

Stages and Pages

- Opening Game Page: The web page that is the starting point for all Flamenco+ searches in a given Flamenco+ Instance. It also provides a count of items by facet value for the entire database.
- Middle Game Page: A web page that lists the identifiers (document number, report number, etc.) of the items that share the value(s) of the currently-selected facet(s).
- End Game Page: A web page that displays the values of the facets and attributes for a single item selected from a Middle Game page.

5

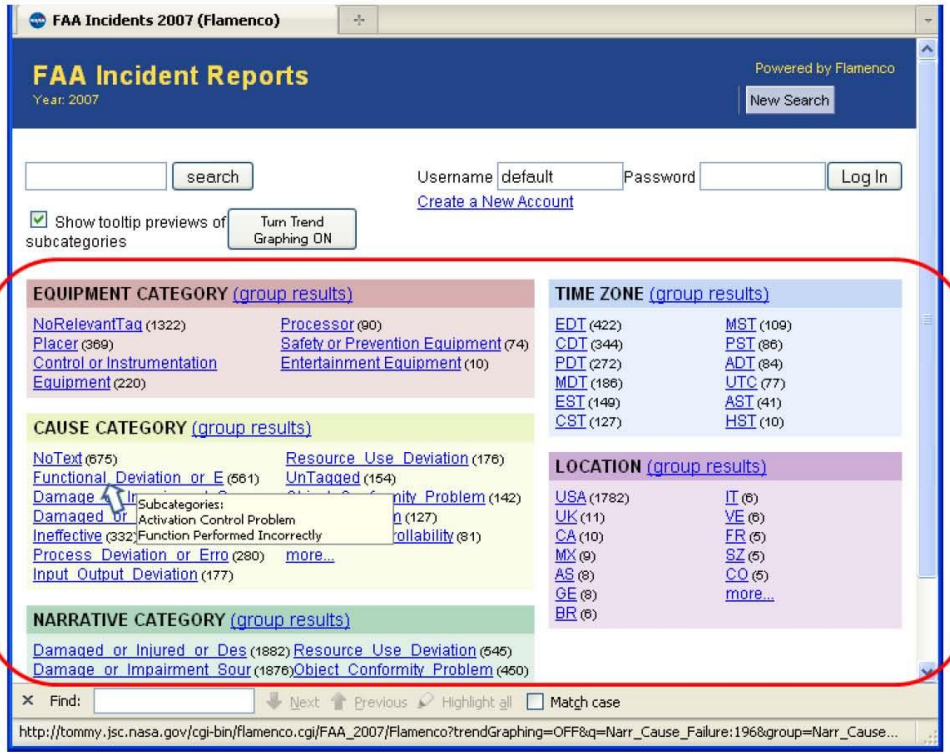
Choose a Starting Point for a Flamenco Search

The Opening Game page provides three alternative starting points for discovery:

1. Area of Primary Concern: For this tutorial, we chose to start with the "*Cause Category*" facet based on the assumption that a person concerned with air traffic safety might be primarily interested with malfunctions responsible for causing accidents, so we start the search by selecting the "*Cause Category*" value: Functional Deviation or E for this tutorial. A person concerned with safety issues in specific region of the country might instead start with one of the possible values of the "**Time Zone**" facet.
2. Frequency of Occurrence: Numbers in parentheses to the right of the facet values on the Opening Game page indicate how many items have that facet/value combination. Especially for problem trend analysis, a facet and value combination with the most items may be a good place to start.
3. Keyword Search: When none of the facet/value combinations on the Opening Game page appear to be suitable, you can start by entering a word in they keyword search form in the upper left corner of the Opening Game page. The match must be exact, so if for instance, the word "wing" returned no matches, try "wings" instead.

6

Opening Game Page for FAA Incident Reports



7

Bullet Symbols Used in This Tutorial

- ❑ Indicates an action you should perform to follow along with the tutorial.
- ➢ Indicates a consequence of a performed action.
- ❖ Indicates other important information more indirectly related to the action being performed.

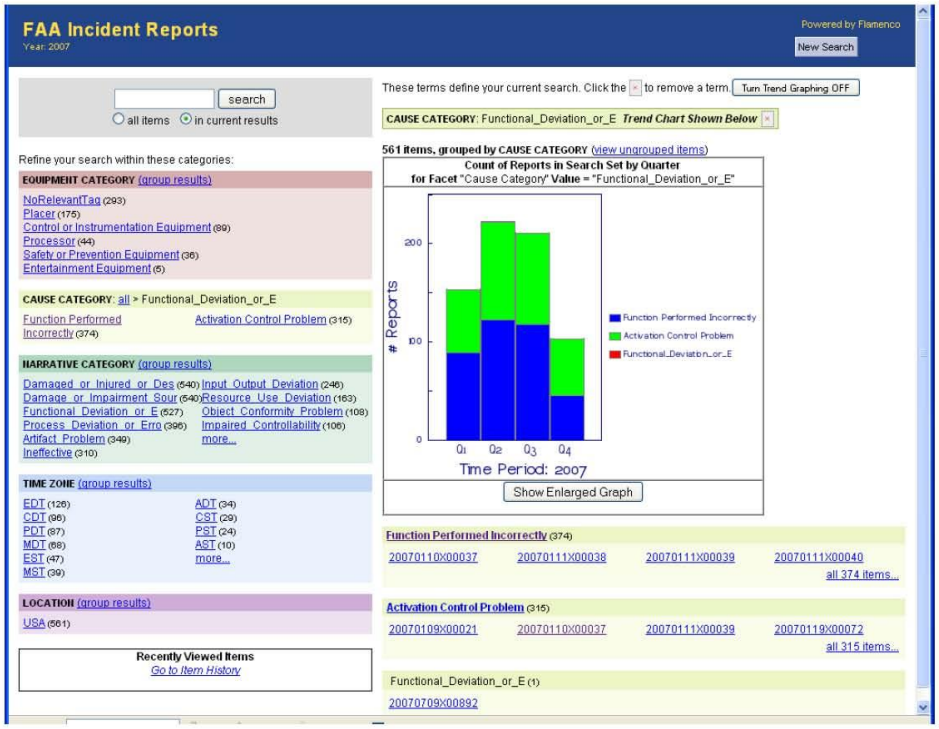
8

Selecting the First Facet Value

- ❖ Opening Game Page Area outlined in red encloses the 5 facets defined for the FAA reports items. The facet names are in bold font.
- ❑ Click on the button at the top left labeled "Turn Trend Graphing On"
- ➢ This enables the display of a "Trend Graph" on any Middle Game page following the selection of any facet value.
- ❖ The facet values listed under the facet name are hyperlinks to Middle Game pages.
- ❖ Note that under the "*Cause Category*" facet, the number in parentheses to the right of the facet value: Functional Deviation or E indicates it is the most prevalent category of incident for which a cause was determined.
- ❑ Look under the facet "*Cause Category*" and hover the mouse pointer over the hyperlink labeled Functional Deviation or E.
- ➢ The tooltip displays the possible top-level values of the Functional Deviation or Error category.
- ❖ Each top-level value is the root of a separate hierarchies of value categories. These categories may have "child" categories that are more specific.
- ❖ Any item whose facet value is a child or descendant of a selected facet value category is also a member of the set of items for the parent category. See Section 3 of the User Guide for more details on hierarchies.
- ❑ Click on the hyperlink Functional Deviation or E under "*Cause Category*".
- ➢ The Middle Game page is displayed for the subset of items in the database with the selected value of the "*Cause Category*" facet. The various parts of this page are shown in the diagram on Slide 11 and will be explained in more detail throughout this tutorial.

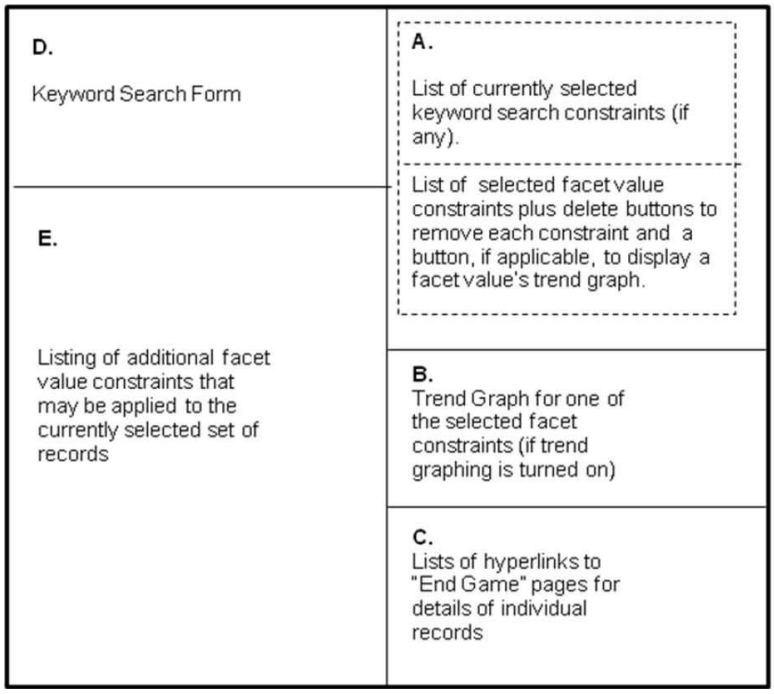
9

Initial Middle Game Page or Facet "*Cause Category*" Value Functional_Deviation_or_E



10

The Areas of the Middle Game Page



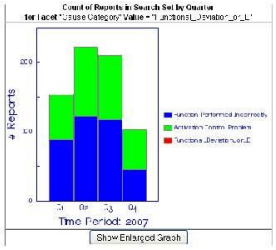
11

Area A of the Middle Game Page

- Area A lists the current facet value and/or keyword selections that were made either on the Opening Game page or on a previous Middle Game page resulting in the subset of items listed in Area E.
- Clicking the button with the red "X" to the right of a search constraint (or filter) causes that constraint to be removed. Since there is only one such constraint here, clicking on it will simply return you to the Opening Game page that shows the distributions of all items unconstrained.

12

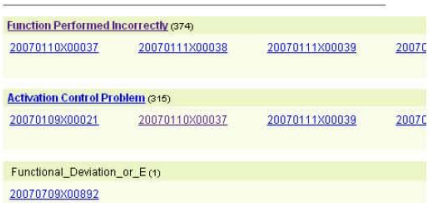
Area B of the Middle Game Page



- Area B contains a bar chart showing the distribution of items over time by quarter or by year, depending on the span of time covered for the selected set of records.
- The colored bands within each bar indicate the number of items for each child category of the currently selected facet value.
- For constraint selections having no subcategories, all bars have a solid blue color.

13

Area C of the Middle Game Page



- Area C lists the identifiers for the first few items in the set of items satisfying the facet value and/or keyword search constraints listed in Area A.
- Items are listed in ascending order by their item IDs. (For the FAA database, the item IDs are the incident investigation report numbers).
- Each item identifier is a hyperlink to the End Game page that describes details about the item's attributes and facets.
- If the currently selected facet value has child categories, the items will be grouped under the child categories (for the Middle Game page on Slide 10, this is the case).
- For subcategories having a large number of items, a link on the far right of the value hyperlink, such as the one labeled all 374 items... to the right of Function Performed Incorrectly, accesses the entire set of items under that child category.

14

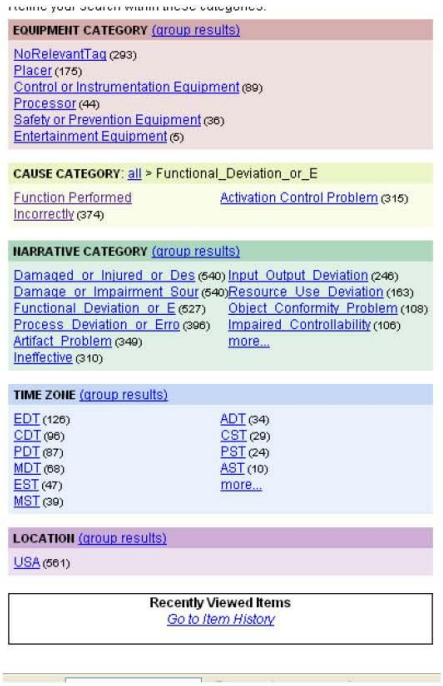
Area D of the Middle Game Page

- Area D contains the form for entering a word or words to further narrow the selection of items.
- If an item contains the words entered, they will be in the item's text *attributes* and are not facet values.
- All words that you enter must match exactly.

15

Area E of the Middle Game Page

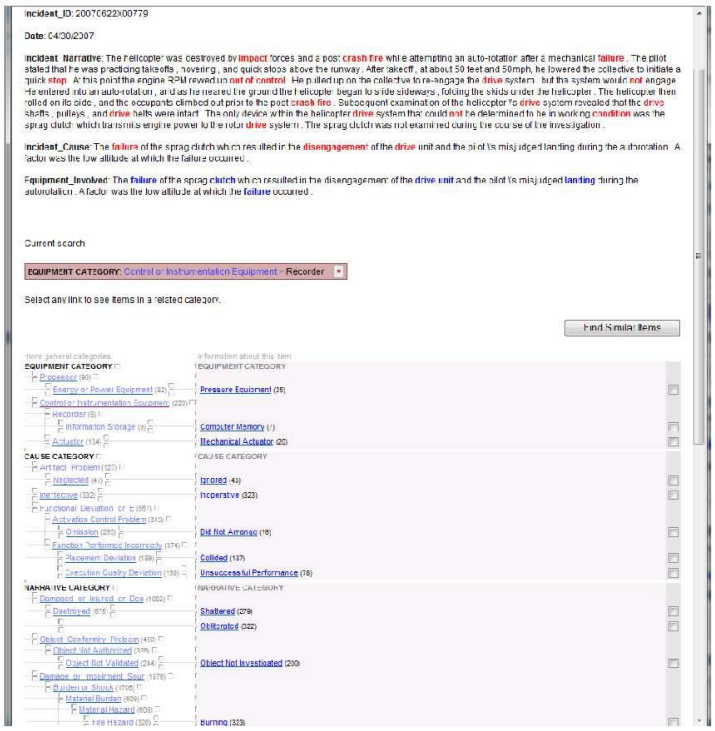
- Area E lists the values of facets that can further narrow the set of selected items. Note that the "*Cause Category*" facet is still listed as a selection, but that the set of possible values are all child categories of the previously selected value: Functional_Deviation_or_E.
- Also in Area E note that USA is the only value listed for the "*Location*" facet because there are no items in any of the other locations in the selected subset.
- The item count to the right of each facet value is less than the count for the same facet value on the Opening Game page because the counts refer only to the selected subset of items on the Middle Game page.



16

The End Game Page

- Inspect a single report
- Refine the search by using attributes tagged in that report

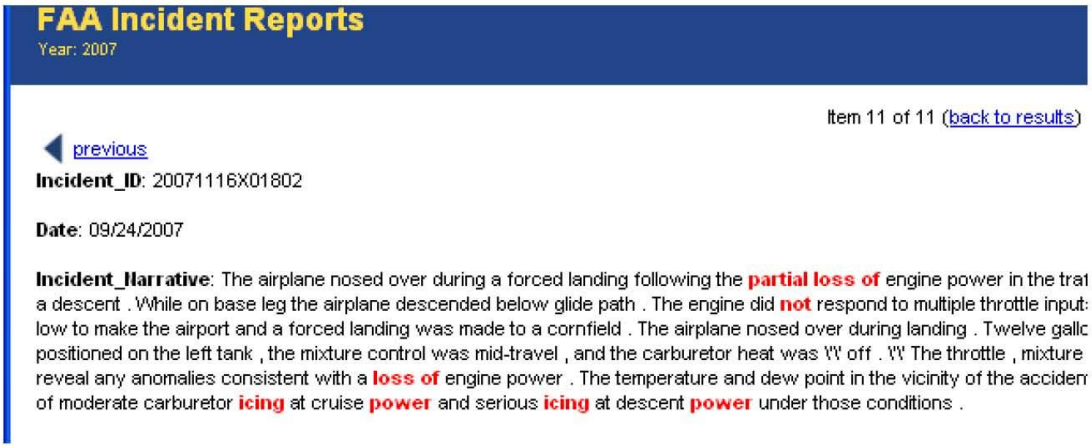


17

Areas of the End Game Page

Upper part of End Game page:

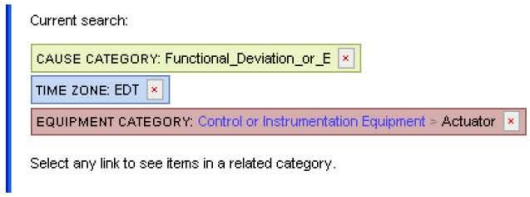
- List of data record attribute names in **bold** followed by their values.
- The first attribute listed is the item identifier (such as a document number or document section number.
- Colored font indicates words used to infer facet values using natural language processing and semantic tagging (see Section 1.2 of User Guide for more details on semantic tagging)



18

Areas of the End Game Page

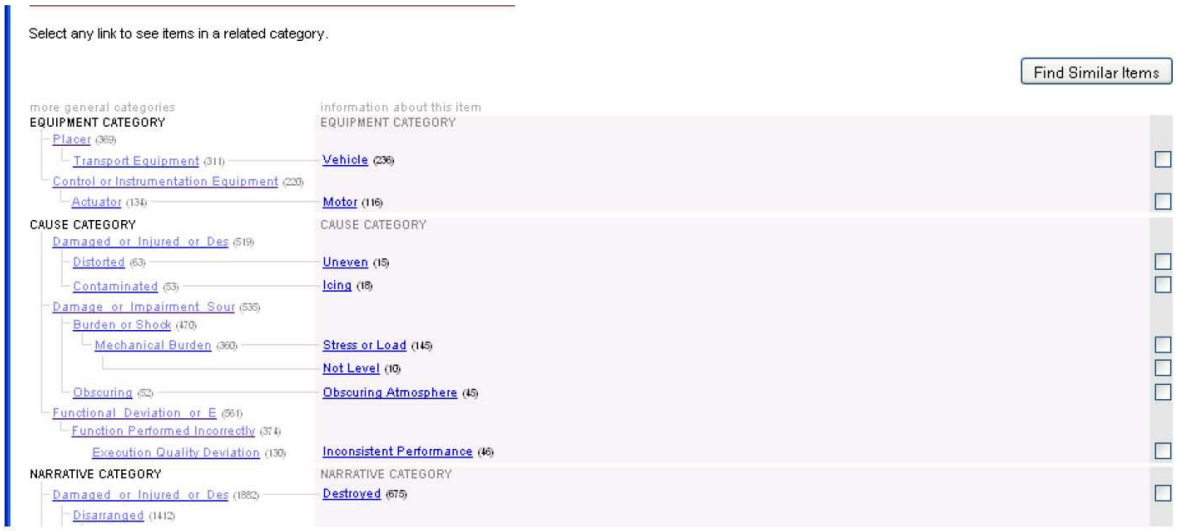
Middle part:
List of search constraints on the Middle Game page from which the item was taken (similar to Area A of Middle Game page but without the "Show Trend Graph" button)



19

Areas of the End Game Page

Lower part:
Right side: the facet check boxes and "Find Similar Items" button previously described.
Left side: Tree representations of the paths through facet value hierarchies leading to each of the item's facet value. (See Section 3 of User Guide for more details on value hierarchies)



20

Selection Strategies

- Four Ways to Refine a Search
  - Option 1: Add New Facet Constraints
  - Option 2: Narrow the search to a subcategory of the value of a currently selected facet
  - Option 3: Using an Item's End Game page to find similar items
  - Option 4: Match words appearing in an item's attributes on an End Game page
- Remove Constraints on a Selection to Broaden a Search

21

How to Refine a Flamenco Search

There are four main ways to refine a search. They can be repeatedly performed in any order:

1. Add a new facet to your current facet constraint selections on a Middle Game page.
2. Narrow the search to a subcategory of the current value of a currently selected facet on a Middle Game page.
3. "Find similar items" on an item's End Game page (a page showing the information on a single Flamenco item), based on the values that the item has for one or more its facets.
4. Match words appearing in an item's attributes on an End Game page.

❖ For alternatives 1, 2, and 3, choose facets according to either area of concern or frequency of occurrence, just as when you choose the starting point for a search.

22

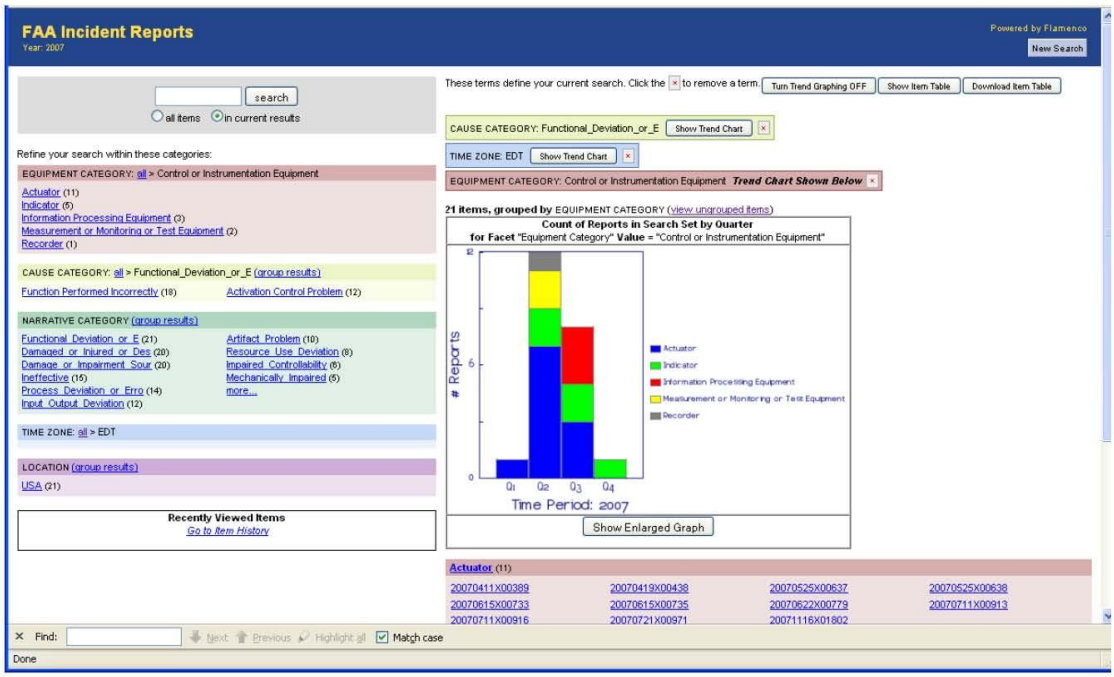
Refining a Search Option1:

Add New Facet Constraints

- ❑ In Area E of the Middle Game page in Slide 10, click on the "*Time Zone*" value: EDT
- ➢ A second Middle Game page is displayed and the item counts adjacent to facet values has changed again.
- ➢ The Trend Graph that showed the distribution over time for the "*Cause Category*" constraint and its subcategories is also replaced with the graph for the EDT time zone. The bars are all a solid blue color because time zones have no subcategories.
- ❑ Again in Area E on this new Middle Game page, click on the "*Equipment Category*" facet value: Control or Instrumentation Equipment.
- ➢ A third Middle Game page is displayed as shown on the next slide.
- ➢ The Trend Graph now shows the distribution over time for the "*Equipment Category*" value of Control or Instrumentation Equipment and its subcategories. The bars on the Trend Chart are once again banded, this time for the subcategories of "*Equipment Category*" .
- ➢ The hyperlinks in Area C to the End Game pages for the selected subset of items are grouped by subcategory of that facet value.
- ❖ You could have arrived at the same Middle Game page if you had selected the value of the "*Equipment Category*" before selecting the "*Time Zone*" value.

23

Middle Game Page for constraints on 3 different facets



24

Refining a Search Option 2:

Narrow the search to a subcategory of the value of a currently selected facet.

- ❑ Click on the "New Search" button to go back to the Opening Game page.
- ❑ Again click on the "Cause Category" facet value: Functional Deviation or E.

➢ This brings you back to the first Middle Game page of Slide 10. You could have arrived at the same place either by using your browser's back-page button or clicking the red X buttons to the right of the names of the "*Time Zone*" and "*Equipment Category*" facets in Area A of the Middle Game page in Slide 24.

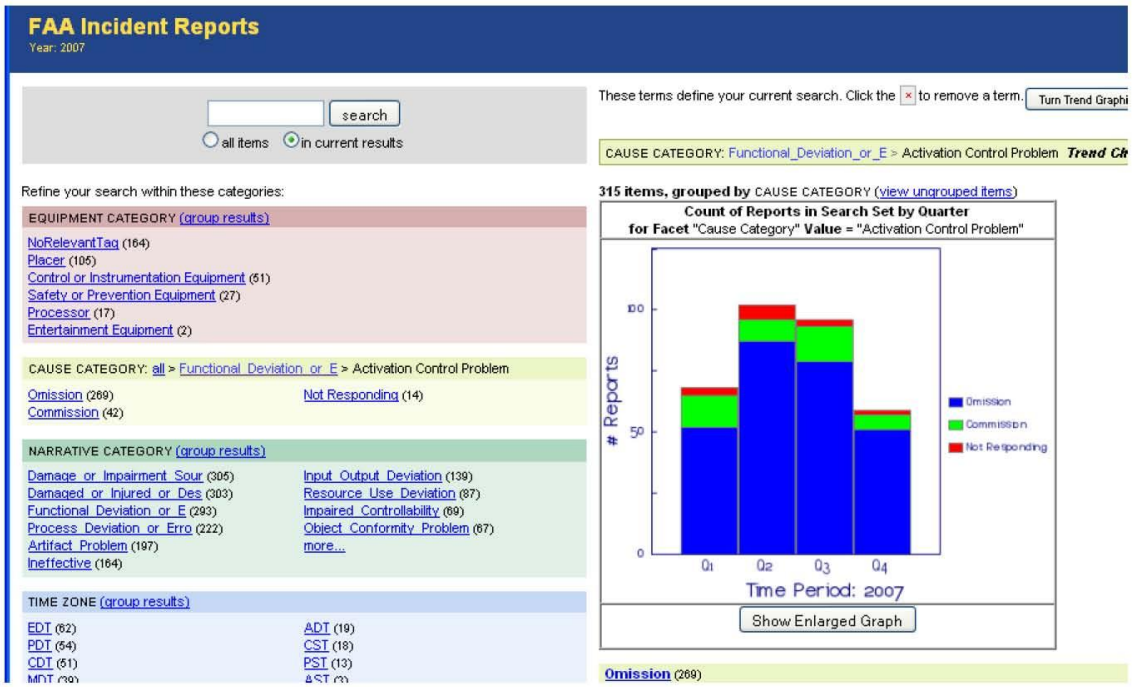
❖ On the Middle Game page of Slide 10, notice that the values of the "*Cause Category*" facet are not the same as they were on the Opening Game page. Here, the only choices are the two child categories of Functional Deviation or E that were displayed by the tool tip on the Opening Game page.

❑ In Area E, click on the hyperlink for the Activation Control Problem subcategory of the "*Cause Category*" facet.

➢ The Middle Game page is now displayed as on the next slide, with the values listed under "*Cause Category*" now limited to the child categories of Activation Control Problem (the "grandchildren" of Functional Deviation or E)

25

A Middle Game page for a lower-level value in a hierarchy of values



26

Refining a Search Option 3:

Using an Item's End Game page to find similar items

❑ Use the browser's Page-Back button or the "New Search" Flamenco button to redisplay the Middle Game page in Slide 10, which shows all items having a "Cause Category" value of Functional Deviation or E.

❑ In Area C, click on the hyperlink for the lone item in the bottom row having the report ID: 20070709X00892

➢ The End Game page for the selected report opens as shown on the next slide.

❖ The lower portion of the page shows the facet values for the selected item, each having the count of items having the same value for that facet in the selected set of items.

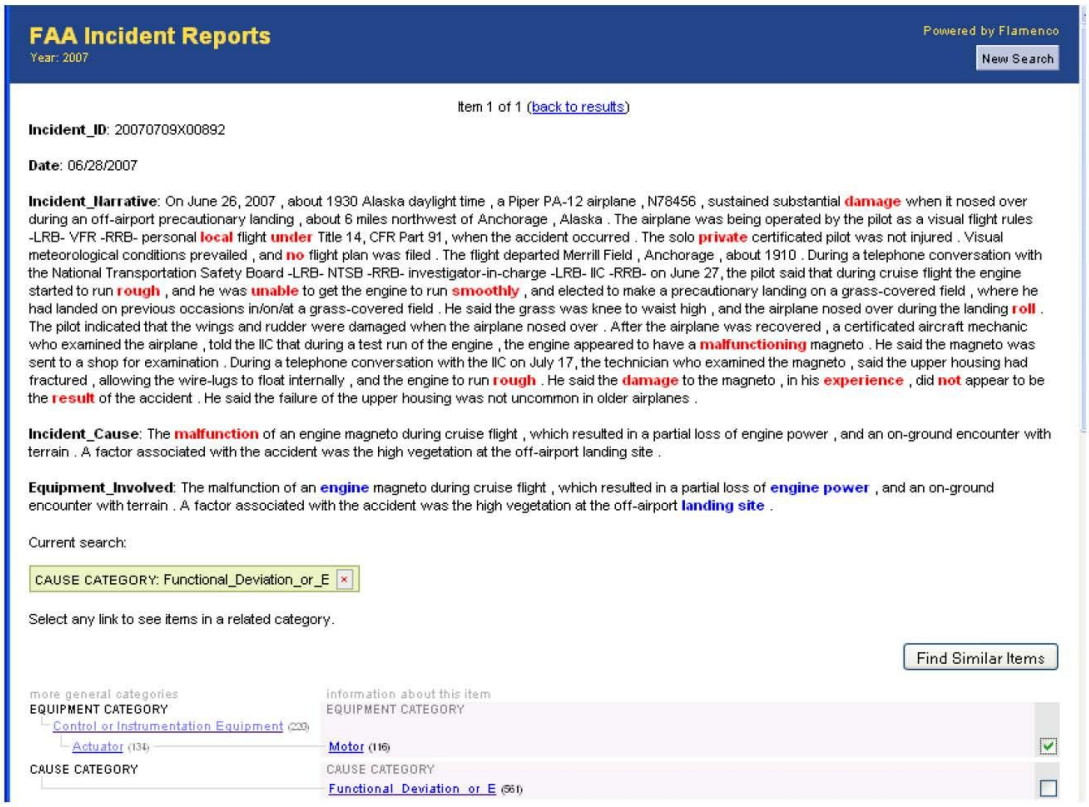
❑ Click on the checkbox to the far right on the web page on the line for the "*Equipment Category*" facet value: Motor.

➢ The button labeled "Find Similar Items" on the resulting End Game page now shows the count of items (116) that have the facet value that you checked off.

[Continued]

27

End Game page showing the item's attributes at top and the facet values at bottom



28

Refining a Search Option 3 [continued]:
Using an Item's End Game page to find similar items

❑Click on the checkbox to the right of the facet "*Cause Category*" value Functional Deviation or E.

➢ Two facet values have now been checked off and the button labeled "Find Similar Items" now shows a count of 39 items – the number of items having the same values for both the "*Equipment Category*" *and* "*Cause Category*" values as the selected item.

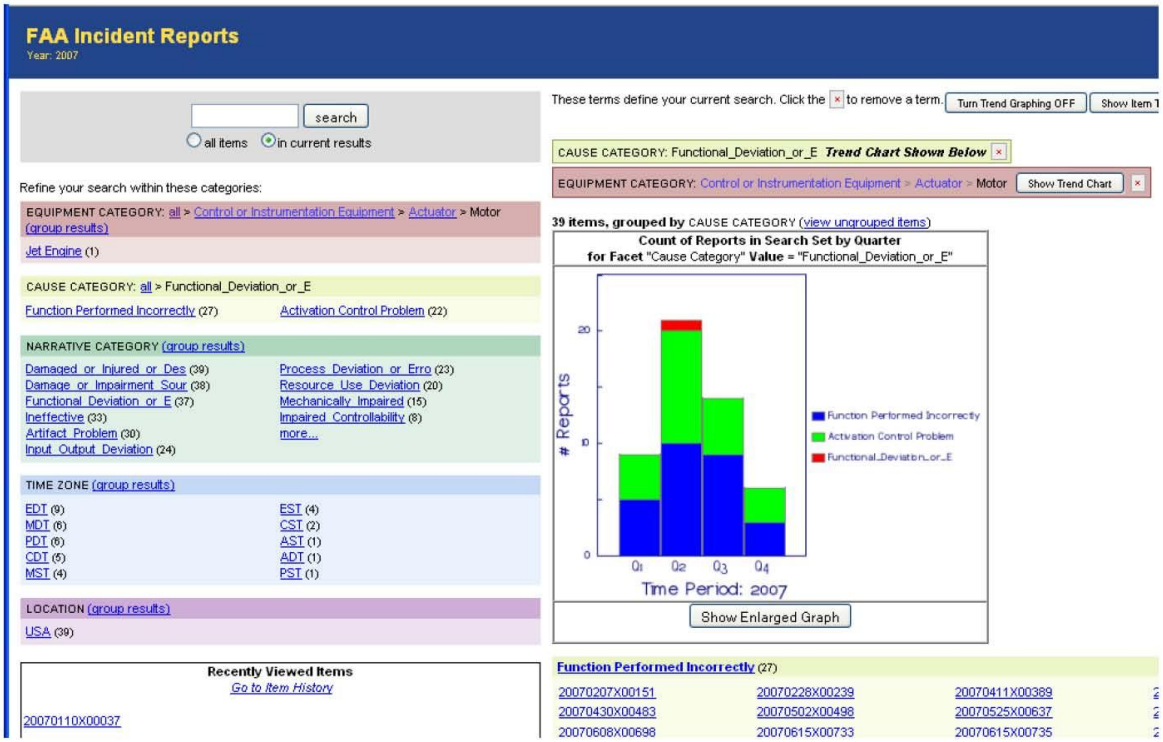
❑ Click on the "Find Similar Items" button.

➢ Flamenco displays the Middle Game shown on the next slide.

❖ This Middle Game page could also have been generated by Option 2 (selecting the first facet value from the Opening Game page and subsequent facet values from the resulting Middle Game page). But the advantage of Option 3's "bottom up" search method is that it may reveal combinations of facet values of interest on an End Game page that are not evident using Option 2's "top down" search method.

29

Middle Game page generated by selecting two facet constraints from an End Game page



30

Refining a Search Option 4:

Match words appearing in an item's attributes on an End Game page

The subset of items in a Flamenco database that match one or more words can be retrieved. All words used for the search must be contained in an item to satisfy the match constraints.

Rather than guessing what words a database might contain, it is better to examine the text attributes of a few items retrieved by apply some combination of the facet-based search methods described previously.

❖Note that in the End Game page on Slide 28, the word "magneto" figures prominently.

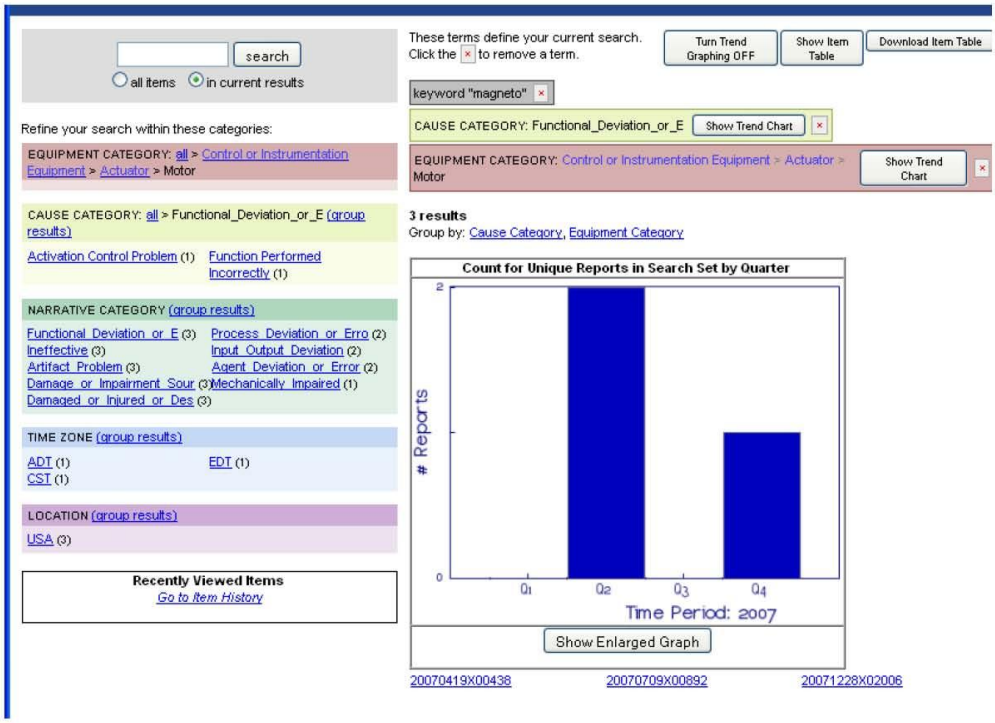
❑ In your web browser, go to the last Middle Game page (previous slide), in which two facet-value constraints have already been imposed.

❑ Enter the word "magneto" in box in the upper left corner of the Middle Game web page (Area D on Slide 11) and click on the "search" button to the right of the word entry field.

➢ A new Middle Game page as shown on the next slide is displayed listing the three FAA incident reports in the previous subset containing the word "magneto" as well as having the values specified for the "*Cause Category*" and "*Equipment Category*" facets.

31

A small set of items with one word match constraint added to two previous facet value constraints.



32

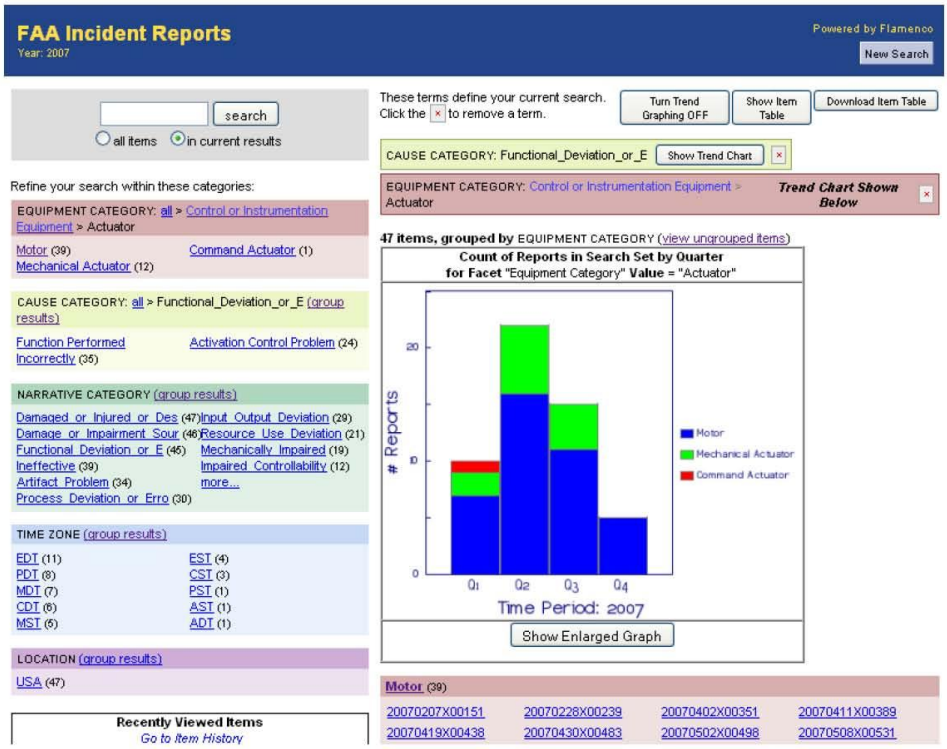
Removing Constraints on a Dataset

All constraints can be removed by clicking the "New Search" button at the top of Area A of a Middle Game page, which returns you to the Opening Game page. However, there are two ways to partially relax constraints that we step through below:

- ❑ With your web browser, go to the Middle Game page shown on last slide with the two facet constraints and one word match constraint. To work with a larger set of items:
- ❑ Click on the red X next to the label keyword "magneto"
  - ➢ The word match constraint is removed completely from the Middle Game page.
- ❑ Click on the hyperlink Actuator in the Area A label:
  "Equipment Category: Control or Instrumentation Equipment > Actuator > Motor"
  - ➢ The Middle Game page now has two facet value constraints as shown on the next slide. The word match has been removed entirely and the "*Equipment Category*" has been broadened.
  - ➢ The Trend Chart in Area B and the End Game hyperlinks in Area C are now grouped into subcategories of the Actuator value of the "*Equipment Category*" facet because that was the last constraint imposed on the set of items.
- ❖ A Trend Chart showing the distribution of the "*Equipment Category*" value over time can still be viewed for the selected set as will be discussed next.

33

Middle Game page after relaxing two constraints on the previous Middle Game page



34

Views and Outputs

- Three Ways to Graph a Selected Subset of Items
  - By changing the facet constraint "in focus" on the Middle Game page
  - By selecting a non-constraining facet value in Area E of the Middle Game page
  - By viewing items ungrouped
- Item Table Web Pages
- Item Spreadsheet Output

35

Three Ways to Graph a Selected Subset of Items (First Way)

By changing the facet constraint "in focus" on the Middle Game page.

❑ On the previous Middle Game page in the upper right (Area A in the diagram of Slide 11), click on the button labeled "Show Trend Chart" to the right of the label Cause Category: Functional_Deviation_or_E.

➢ As shown on the next slide, the facet "*Cause Category*" is now "in focus" and the Trend Chart in Area B of the Middle Game page once again shows a distribution of items versus time for the subcategories of that facet value, and the hyperlinks to End Game pages are once again grouped by the subcategories of Functional_Deviation_or_E.

➢ A "Show Trend Chart" button now appears to the right of the Area A label:

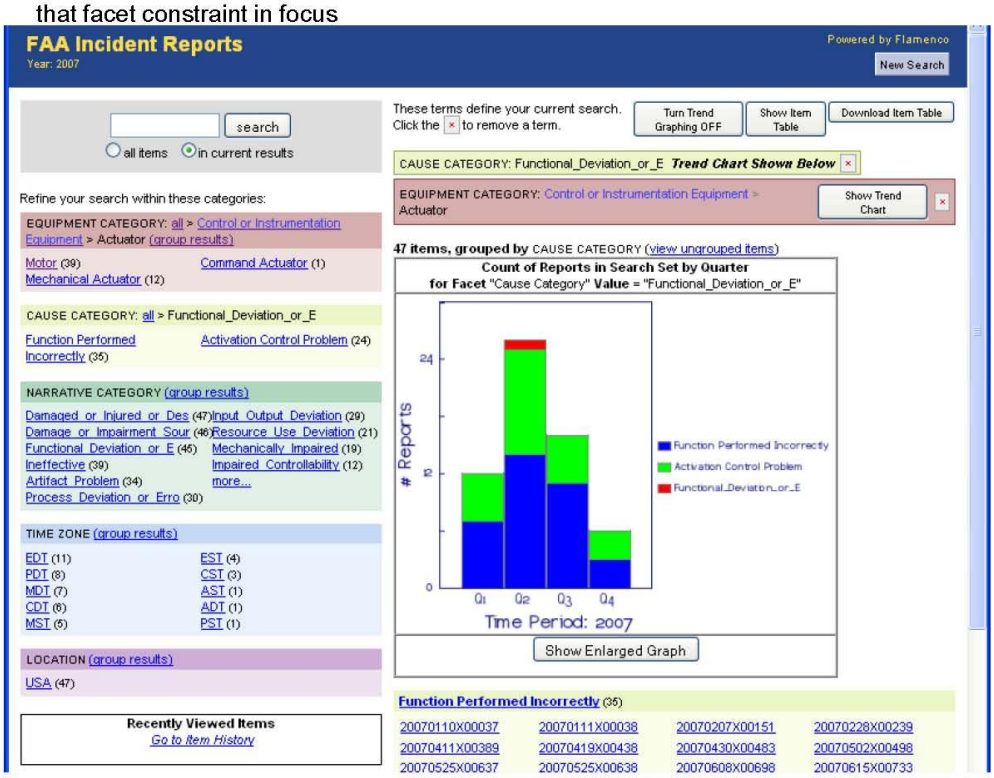
"Equipment Category: Control or Instrumentation Equipment"

This button permits the "*Equipment Category*" facet constraint to be put back in focus if desired.

❖ Changing the focus of the Middle Game page does not change the selected subset of items.

36

Middle Game page after clicking button next to the "Cause Category" label" to put that facet constraint in focus



37

Three Ways to Graph a Selected Subset of Items (Second Way)

By selecting a non-constraining facet value in Area E of the Middle Game page.

❑ On the lower left side of the last Middle Game page (Area E on Slide 11), click on the hyperlink (group results) to the right of the facet name "*Time Zone*."

➢ The bars on the Trend Graph now represent the distribution over time for Time Zones.

➢ The hyperlinks to item End Game pages in Area C are now grouped according to Time Zone.

➢ Both of the two facet constraints ("*Cause Category*" and "*Equipment Category*") are still in effect but neither is in focus. A "Show Trend Chart" button now appears next to both facets in Area A to return the focus to either of them if desired.

❖ Note on the Opening Game page on Slide 7 that there is also a (group results) hyperlink next to each facet name. By clicking on one of those links, a Middle Game page appears that shows the distribution of items in the entire database for the selected facet's top-level values.

38

Three Ways to Graph a Selected Subset of Items (Third Way)

By viewing items ungrouped.

Important: An item may have more than value for a given facet constraint, and more than one of those values may be subcategories of the facet constraint currently in focus. This means that the item will be counted more than once when constructing the Trend Graph of Area B and may appear more than once in the End Game hyperlink listings of Area C. Viewing the items ungrouped shows a Trend Graph distribution in which no item is counted more than once.

❑ Click on the hyperlink (view ungrouped items) as shown on next slide on the last Middle Game page (Area A just above the Trend Chart).

➢ The bars on the Trend Graph in Area B are now all a solid blue color, with no item counted more than once.

➢ The hyperlinks to item Eng Game pages in Area C are now ungrouped, and are listed in alphabetical order of item ID regardless of any facet constraint subcategory.

❖ If any facet value that has no subcategories is put into focus, the distribution will be exactly the same as when the (view ungrouped items) option is chosen. Similarly, if the last operation was to add a word match constraint,, there are no subcategories to display and the solid blue graph is therefore displayed. Each item is counted only once in either case and the graph bars are the same solid blue color.

39

Location of “view ungrouped” hyperlink

40

Viewing Items in Tabular form

On the last Middle Game page (Slide 37), there are two buttons to the right of the "Turn Trend Graphing Off" button at the top of Area A (the upper right of the page). These provide two alternative forms of tables, each with its own advantages and disadvantages:

- Clicking the "Show Item Table" button generates a new web page that lists items alphabetically by their item ID. The table includes all special fonts, highlighting , and any hyperlinks that appear on the items' individual End Game pages. As shown on the next slide, the facet constraints are listed in the upper left of the page. At the upper right is a count of items in the table and a list of index numbers that are hyperlinks for display of additional pages of the table in cases where there are more than 500 items in the selected subset (not the case here).

- Clicking on the "Download Item Table" button initiates the downloading or display of a tab-separated value file in a spreadsheet as shown on Slide 43. While the End Game fonts and hyperlinks are not preserved in the spreadsheet, other spreadsheet operations can be performed on the downloaded items that cannot be performed on the static web page version of the table. Also, since all rows are printed on one spreadsheet line, more items can be seen at one time than in the web page version of the table. The first rows the spreadsheet show all of the facet and word match constraints that produced the subset of items listed in subsequent rows of the table.

41

Web Page table of items showing search constraints and first row of data

FAA Incident Reports
Year: 2007

Powered by Flamenco
New Search

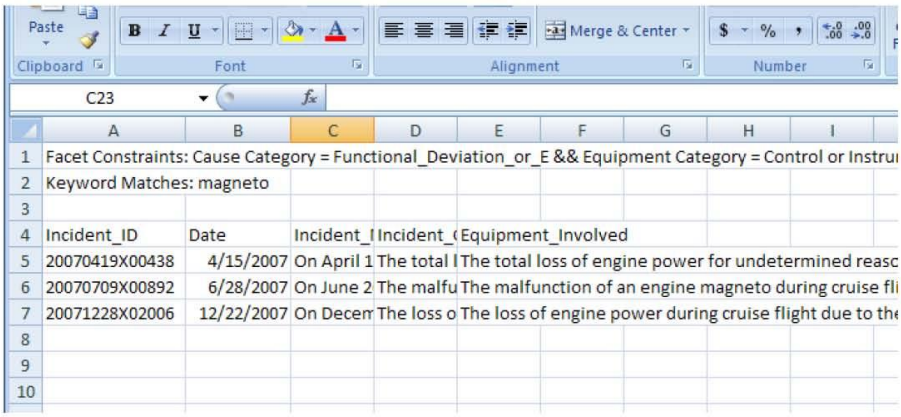
Facet Constraints
- **Cause Category** = "Functional_Deviation_or_E"
- **Equipment Category** = "Control or Instrumentation Equipment"

Keyword Constraints
- magneto

Items 1 to 3 of 3 results

Incident_ID	Date	Incident_Narrative	Incident_Cause	Equipment_Involved
20070419X00438	04/15/2007	On April 13, 2007 , about 1110 eastern daylight time , a Consolidated Aeronautics , Inc. , Lake LA-4-200 , N8543W , registered to N8543W , Inc. , experienced a **loss of** engine power and collided with power lines then the ground during a forced landing **shortly** after takeoff from the SarasotaVBradenton International Airport , Sarasota , Florida . Visual meteorological conditions prevailed at the **time** and no flight plan was filed for the 14 CFR Part 91 personal flight from SarasotaVBradenton International Airport , Sarasota , Florida , to Bartow Municipal Airport , Bartow , Florida for a seaplane rally . The airplane was substantially damaged and the private-rated pilot sustained minor injuries , while the passenger was seriously injured . The flight was originating at the time of the accident . The pilot stated that he performed a preflight inspection of the airplane using the checklist . The main fuel tank which was full was checked for contaminants ; none were found . He did **not** report any discrepancies associated with the engine start , and after doing so , taxied using only the mechanical fuel pump to runway 04 . He performed an engine run-up and each magneto drop was approximately 100 rpm . He turned on the auxiliary fuel pump , and noted the fuel pressure was OK . He estimated the airplane was on the ground approximately 10 minutes from the time of engine start to the moment he began the takeoff . The flight was cleared to takeoff , and he gradually applied full power , noting that the airplane accelerated satisfactorily . He rotated at 60 mph and after obtaining a positive rate of climb , retracted the landing gear . The aircraft accelerated to only 65 mph **instead of** the usual 80 mph , and the pilot recognized that the airplane performance was low . The tower controller asked him if he was experiencing a problem , and he verified the mixture , throttle , and propeller controls were full forward . The engine then experienced a near **total loss** of engine power with resulting pitch-up -LRB- normal -RRB- . He lowered the nose to **maintain** airspeed -LRB- 60 mph -RRB- , and after recognizing that he would be **unable to land** on the airport , maneuvered the airplane to land on a nearby street . While the left wing descending when the flight was approximately 30 feet above ground level , the left wing collided with a power line . The airplane then rotated approximately 90 degrees to the left , and impacted onto the road . Bystanders helped the passenger	The **total loss** of engine power for undetermined reasons during initial climb after takeoff , resulting in a forced landing , and an in-flight **collision** with transmission wires and	The **total** loss of **engine power** for undetermined reasons during **initial** climb after takeoff , **resulting** in a forced **landing** , and an in-flight **collision** with transmission wires and terrain .

Excel spreadsheet of same table as on previous slide



	A	B	C	D	E	F	G	H	I
1	Facet Constraints: Cause Category = Functional_Deviation_or_E && Equipment Category = Control or Instru								
2	Keyword Matches: magneto								
3									
4	Incident_ID	Date	Incident_I	Incident_(	Equipment_Involved				
5	20070419X00438	4/15/2007	On April 1	The total l	The total loss of engine power for undetermined reasc				
6	20070709X00892	6/28/2007	On June 2	The malfu	The malfunction of an engine magneto during cruise fli				
7	20071228X02006	12/22/2007	On Decem	The loss o	The loss of engine power during cruise flight due to the				
8									
9									
10									

Appendix F. The Flamenco+ Faceted Browser User Guide for Trend Analysis

The Flamenco Faceted Browser

User Guide for Trend Analysis

Engineering Directorate
Software, Robotics, and Simulation Division

Date: September, 2011

National Aeronautics and Space Administration

Lyndon B. Johnson Space Center

Houston, Texas 77058

Contents

1 Overview of Faceted Browsing for Trend Analysis

The conventional way to explore and search in data on the web is to follow a set of hyperlinks from a beginning page to a page (or record) of interest. In trend analysis, the characteristics of a group of records sharing common features are generally of greater interest than individual records. Groups of related records can be accessed via conventional web page hyperlinks if the web pages are organized as a series of listings or menus leading to to a final list of records that share some property in common with each other. The properties shared by the records listed on the final menu are essentially fixed by the web site's design scheme.

However, retrieving a set of records that are related to each other in a predetermined way is not always adequate for trend analysis. For instance, problem reports concerning "batteries," that have "expired" may be of great interest over some time period in which there is a large proportion of problem reports concerning expired batteries. At other times, the more prevalent problem with batteries could be manufacturing defects rather than their use past their expiration date. In either case, an analyst would want to monitor the trends for the specific types of equipment and problems of greatest recent concern to see if the problems' occurrence is increasing or descreasing over time.

Faceted browsing is now commonly used to provide more flexibility as to what properties you can select to characterize a set of records: You specify the value of one or more properties (or record fields) and the browser application generates web pages that list the records whose property values match the selected value constraints. The properties that serve as constraints for record set selection are referred to as "facets," giving this style of browsing its name.

In general, not all of the record properties are facets; the browser application designer must choose the facets, and the choices are limited to those properties that have a finite number of possible values. The database designer selects as facets those properties deemed most likely to be useful to the end user. For example, in a problem report database the fields describing equipment type and problem type would likely be more important in characterizing a group of records for trend analysis than the field that states the name of the report author, which the database designer would be less likely to choose to be a facet.

Faceted browser applications generally let the analyst add new facet value constraints or remove old ones during searches for trends in the data. In essence, faceted browsing is a series of database queries, each query specifying a new property value constraint to narrow the search as shown in Figure 1.

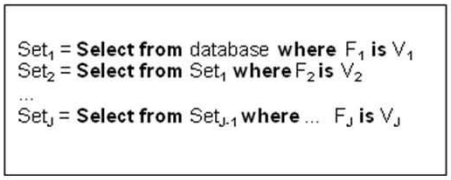


Figure 1: A series of queries equivalent to the process of refining the set of records being examined in a faceted browser application.

A clause specifying that "F_i **is** V_i" is referred to as a "facet value constraint" in this guide. The contents of the final set of records is independent of the order in which the facet value constraints are applied.

Every new set in the sequence above is retrieved by clicking on a hyperlink on the page displaying the results of the previous selection. The final set of records, Set_J, would be the set of all records in the database having the values specified for the *J* facets specified: F_1 through F_J. The same final set of records is equivalent to the set retrieved in the single query:

Set_J = **Select from** database **where** F_1 **is** V_1 and F_2 **is** V_2 **and** ... F_J **is** V_J

1.1 The Flamenco Browser Application Basics

Flamenco is the faceted browser application that has been chosen and adapted for use in trend analysis. Flamenco provides the basic capabilities of faceted browsing augmented with two unique features: hierarchical facet values, and keyword searches that can be mixed in with facet-based searches. A third feature has been implemented expressly for the purpose of trend analysis: bar graphs of the number of records in a chosen set as a function of time.

There are three main types of web pages displayed by Flamenco:

1. The "Opening Game" page (Section 2 in this guide) that is displayed prior to any facet or keyword constraints being chosen. This is the page from which you execute, in effect, the initial database query in the series shown in Figure 1:

   "Set_1 = **Select from** database **where** F_1 **is** V_1"

2. The "Middle Game" page (Section 5 in this guide) that displays the currently selected set of records after the initial query. This page is used to to refine the current data set further by adding new facet value or keyword constraints. Every time you add a new constraint, Flamenco generates and displays a new Middle Game page. The Middle Game page displays a graph showing the number of reports for the selected set of records as a function of time, with the time intervals being years for large periods of time and quarterly for smaller time spans. An example of a Middle Game page is shown in Figure 5.

3. An "End Game" page (Section 7 in this guide) that displays the details of a single record selected from the Middle Game page. The End Game page also contains controls you can use to retrieve records similar to the record shown on the page. An example End Game page is shown in Figure 12 and Figure 13.

In trend analysis, the most important page is often the Middle Game page, because that is the page that provides the information on the characteristics of records as a group, including a bar chart showing the distribution over time of the reports in the set selected.

1.2 Trend Analysis, Flamenco, and the Semantic Text Analysis Tool

Flamenco is the faceted browser application that has been incorporated into the set of tools for created for problem trend analysis. The central component of the toolket is the Semantic Text Analysis Tool (STAT). The toolkit administrator uses STAT to perform an analysis of the natural language text fields in the original problem report records and categorizes the types of equipment and problems referenced in the text.

The categories that STAT assigns to record facets are taken from an Aerospace Ontology. The Ontology is the repository for the hierarchical relationships among the categories, and for the associations between categories and synonymous words and phrases.

STAT records its categorizations in the "tag" fields of a text file in tab-separated value (TSV) format. The tag fields are not part of the original problem report records. The toolkit administrator then runs a Flamenco utility that uses the data text file and other metadata TSV files to generate a SQL database specialized for use by Flamenco.

The Flamenco browser application uses the SQL database and metadata to display:

- The list of facets , which consists not only of STAT tag fields but also selected fields from the original database records that are treated as facets because they have been deemed useful in the analysis process (examples: organizational codes or geographic regions);
- A list of hyperlinks for the allowed values of each facet;
- For each facet value hyperlink, a "tooltip" list of the value's subcategories that will be listed as the allowed facet values on a new Middle Game page if the value hyperlink is activated.

In Flamenco terminology, record fields that are of interest to the analyst but that are not treated as facets are referred to as "attributes." An important attribute for trend analysis is the "Date" field, or equivalent, that provides the date on which a problem report was initiated. Flamenco (as modified for trend analysis) uses this field to construct trend graphs of the frequency of various types of problems or problematic equipment versus time. Other attributes of a records are displayed on the record's End Game page, which is generated when the user activates a hyperlink for the record on a Middle Game page.

The rest of this guide describes Flamenco and its operation in more detail. A database of aircraft incident reports from 2007 serves as the example of how to use Flamenco for problem trend analysis.

1.3 Choice of Browser for Use With Flamenco

It is recommended that you use a Firefox web browser rather than Internet Explorer. Flamenco passes the data for constructing a trend graph as parameters concatenated to the web page URL. Internet Explorer has a limit to the amount of data that can be passed in this fashion. This limitation may lead to truncated data lists being passed resulting in incorrect graphs. There is no limit on the amount of data that can be passed as URL parameters with Firefox.

2 The Flamenco "Opening Game" Page and Facet Selection

Figure 2 shows the initial Flamenco web page for a sample database of incident reports from the Federal Aviation Administration (FAA). This page is called the "Opening Game" in Flamenco terminology. The Opening Game page displays when you start a new search of the database. The red rectangle in Figure 2 encloses the facets and the list of possible values for each facet. The FAA records have 5 facets. The values of the *Time Zone* and *Location* facets were extracted from fields of similar names in the orginal FAA report records. The other three facets are tag fields whose values were derived by STAT from the text in the FAA records.

Every facet value is a hyperlink to a Middle Game page listing the records satisfying that value constraint. To the right of each facet value hyperlink shown in parentheses is the number of records that have that value for the facet. A facet value showing a large number of records can often serve as a good starting point for trend analysis.

After you select a facet value from one of these lists, Flamenco displays a "Middle Game" web page that shows the list of records that satisfy the selected facet value constraint as well as other information on the records as a group. Before describing the Middle Game page, we first describe the concepts of facets, and keyword searches in Flamenco. The Trend Graphing utility is described in the discussion of the Middle Game page, where the Trend Graphs are displayed.

A hyperlink labeled "(group results)" appears to the right of each facet name in Figure 2. Clicking on one of these links opens a Middle Game page that lists the records having each of the facet values for the entire database. If trend graphing is turned on, the Middle Game page will contain a bar chart the distribution of records for each category as a function of time. More often, however, you will want to select a subset of records that have a particular facet value and then optionally display the groupings of records for the possible values of some other facet. Section 5.6 describes how to do this in more detail.

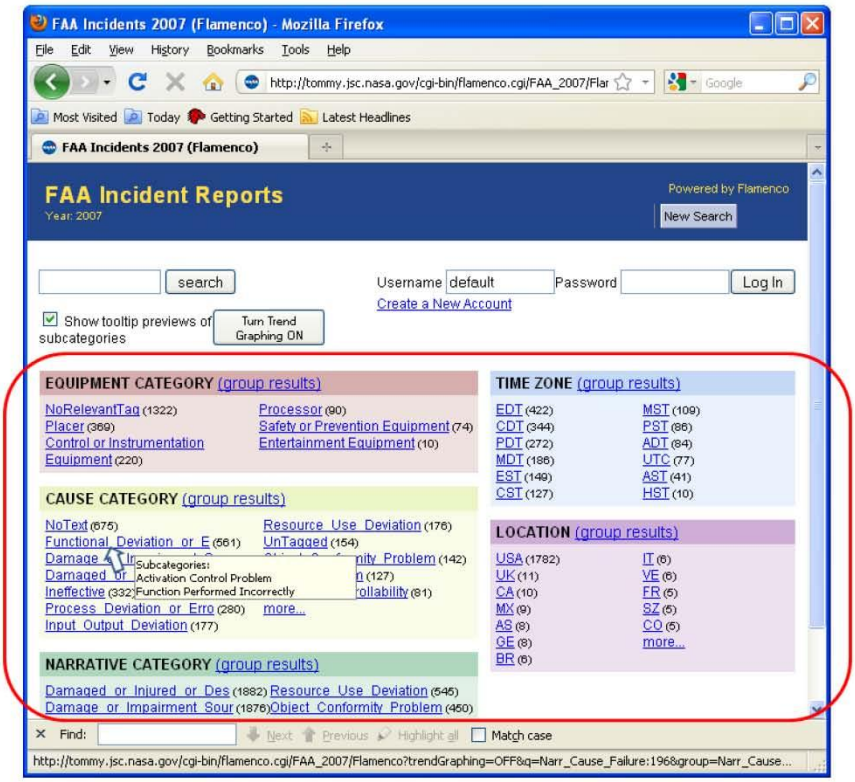


Figure 2: The Flamenco "Opening Game" web page for FAA trend analysis. Facets and their possible values are in the area outlined in red.

In trend analysis, of primary interest is the frequency of occurrence in a dataset of problem types and the equipment involved. For that reason, the possible values for each facet are displayed in descending order according to the number of records possessing that value on the Opening Game page of Figure 2 and in "Middle Game" pages described subsequently. The number of records for each value is shown in parentheses to the right of the value's name.

3 Facets and Facet Values in Flamenco

3.1 Facet Value Hierarchies

A database used by Flamenco may be constructed so that some of the possible values for a facet are organized into hierarchies such as taxonomies, organizational hierarchies, or parts trees. When you select a facet value, *v*, that is a node in a hierarchy, a record will be included in the set retrieved if its facet value is either equal to *v* or is a descendant of *v* in the hierarchy. The list of values from which you make your initial selection for a given facet may be a mixture of "root" nodes of separate hierarchies and non-hierarchical (singleton) values. For facets whose values are hierarchical, a roughly SQL-style query such as those in Figure 1 could be written as:

"Set_j = **Select from** Set_{j-1} **where** F_j **is** V_j **or** F_j **is a descendant of** V_j"

3.1.1 Example of a Facet Value Hierarchy

Figure 3 shows a part of a **Function_Deviation_or_Error** hierarchy that is a possible value in both the ***Cause_Category*** and ***Narrative_Category*** facets. This hierarchy could be of interest to an analyst for further expl;oration due to the large number of reports having this tag (561 reports under the ***Cause_Category*** facet as can be seen in Figure 2). If your initial selection of the ***Cause_Category*** facet's value is **Function_Deviation_or_Err**, Flamenco will retrieve any record having a facet value equal to **Function_Deviation_or_Err** or that is a descendant of that **Function_Deviation_or_Err**, such as the "child" category **Function Performed Incorrectly** or the indirect descendant, **Incorrect Start or Stop**.

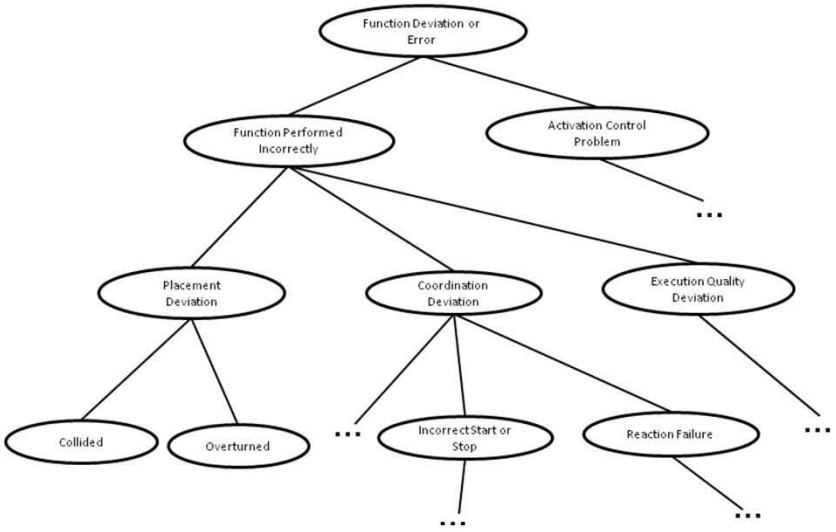


Figure 3: Part of the Function_Deviation_or_Error hierarchy of categories. Ellipses represent additional categories. Categories "Collided" and "Overturned" are leaf categories (i.e., have no subcategories).

3.1.2 Navigating through a Facet Value Hierarchy

On the right-hand side of the Opening Game page in Figure 2, the user has placed the mouse pointer over the **Function_Deviation_or_Err** value hyperlink for the Cause_Category facet, causing the tooltip to display the two immediate subcategories of Function_Deviation_or_Error, which are Function Performed Incorrectly and Activation Control Problem. Using the mouse pointer in this way will always show what the "child" values are for a facet value.

If the analyst clicks on **Function_Deviation_or_Err**, the browser will retrieve the set of all records having that category or any of its descendants (e.g., **Collided**, or **Incorrect_Start_or_Stop**) as a value of the ***Cause_Category*** facet.

Subsequent to the initial selection of a facet value, you can refine your selection by clicking on a hyperlink to one of the subcategories of a facet value such as **Function_Deviation_or_Err**. Flamenco will then retrieve the subset of records in the current set whose value for the ***Cause_Category*** facet contains the most recently selected value, or one of that value's direct or indirect subcategories and exclude records from the selected value's"sibling" categories. The selection process may be repeated, refining the set of records selected until a "leaf" category (i.e., one with no subcategories) is selected. The details of facet value selection are described in Section 5.

The use of hierarchies of values provides to the analyst the advantage of retrieving records that have related values for a given facet rather than having identical values.

3.2 The "More" Hyperlink and Facets with Hidden Values

If the number of allowed values for a facet is too large to fit them all on the Opening or Middle Game web page, the last value hyperlink listed for the facet will be "more...". Clicking on this link will show a web page listing all values for the facet. Because a problem trend analyst is likely to be interested in those facet values associated with the largest numbers of records, Flamenco has been modified to display facet values in descending order of the number of associated records. The hidden values are generally associated with very few records relative to those shown on the Opening or Middle Game page.

3.3 Records with Multiple Values for a Facet

Facets that whose values are extracted directly from a field in the original report records will be single-valued. But a data record may have more than one value for a given facet when the values are tags supplied by STAT. In the case of the FAA database, a single report may have a multivalued facet because that report's text fields may describe more than one type of equipment or problem. Multivalued facets may affect the interpretations of some information on Middle Game pages as described subsequently.

4 Keyword Searches

Flamenco permits you to add keyword constraints to your search, either in combination with facet value constraints or alone. Every record in the set retrieved will then contain at least one occurrence of all keywords specified somewhere in the record in addition to any facet values specified. Flamenco ignores "stop words" so searches on words as "the", "of", etc., will return no results.

Optionally, you may immediately collect records by keyword alone without first selecting any facet constraints. To do this, you enter the search term in the form next to the "search" button in the upper left of the "Opening Game" page in Figure 2. The Middle Game pages also have a keyword search form, also in the upper left region of the page.

Note: The keyword search utility is only available if the Flamenco database adminstrator has created a table of report IDs versus keywords for the report database. The form for keyword entry will not appear on the Flamenco web pages if the database does not contain a keyword table.

The keyword search is not case sensitive, but when doing keyword searches, it must be kept in mind that a record will not be retrieved unless it contains a word that exactly matches every keyword that you enter. The keyword search does not recognize synonyms or variants for the search words entered; the facets to which STAT assigns semantic tags were created for the purpose of dealing with such natural language complications.

You can enter several words into the form at the same time, but only records contains all of the words entered will be retrieved. The End Game pages (Section 7) for individual records may be a good source of keywords. Any word appearing in the record that is not a stop word is guaranteed to return at least that one record when it is entered as a single term for a keyword search.

Performing a keyword search always brings you to a new Middle Game page with information on the records satisfying the keyword constraint, as well as any facet value constraints that are also in effect.

5 The Flamenco "Middle Game" Page

The Middle Game page is the Flamenco web page that generally should be of most interest in trend analysis because it is this page where the characteristics of a set of records can be viewed and evaluated collectively. The Middle Game page provides the analyst a way to examine how the chosen set of records is subdivided into groups by subcategory of any of the facet value constraints that produced the current set of records. Each time the view is changed to show the grouping for a different facet value constraint, a new Trend Graph will also be displayed that shows the distribution of records in each of the facet value's subcategories as a function of time. The facet constraint for which the subcategories and Trend Graph are currently shown is referred to here as the facet constraint that is "in focus."

5.1 Middle Game Page Organization

The Middle Game page, shown in Figure 5, is divided into two columns. The column on the right displays the information on the record set that has already been selected while the column on the left displays the options for further refinement of the current record set. Each time you refine a record set with a new keyword or facet constraint, a new Middle Game page is displayed for the resulting subset and the most recently added constraint is the one that is in focus on that page.

The layout of the information on this page is shown in Figure 4 with the major regions of the page labeled with letters as follows:

A. One row for each facet value or keyword that has been used to produce the current set of records. Any keywords are listed above the facets. Pressing the button labeled with a red "X" to the right of each facet or keyword constraint causes a new Middle Game page to be displayed with that constraint removed. If Trend Graphing is turned on, to the right of each facet constraint that is not the current focus of the Middle Game page is a button labeled "Show Trend Chart" that, if pressed changes the focus to that facet constraint. There is no button in Figure 5 because only one facet has been selected and is automatically the facet in focus on the page. However, in **Figure 6**, where three facet constraints have been selected, there are two such buttons, one for each of the two facets not currently in focus.

B. The Trend Graph, presented as a bar chart (when Trend Graphing is turned on) showing the number of records for each year or calendar year quarter. The size of the time interval depends on the span of time covered in the chart: one year intervals are used for charts covering more

than 5-year time spans, and quarterly intervals for databases covering time spans of less than 5 years. The total height of the each bar represents the total records for the facet value currently in focus. If the facet value has subcategories, each bar is subdivided into color-coded sections representing the number of records falling into each of the subcategories of the chosen facet value. The Trend Graphs are described in more detail in Section 5.7.

C. Rows of hyperlinks to End Game pages, each of which displays the details of an individual record in the currently selected set. There is one row of hyperlinks for each subcategory of the facet value constraint currently being examined, unless the records are being viewed "ungrouped." The document IDs will be displayed in alphabetical order.

D. The form for entering keywords to refine the current set of records.

E. The list of possible facet constraints for refining the current set of records.

These five regions, with reference to the letter designations are discussed in the following sections.

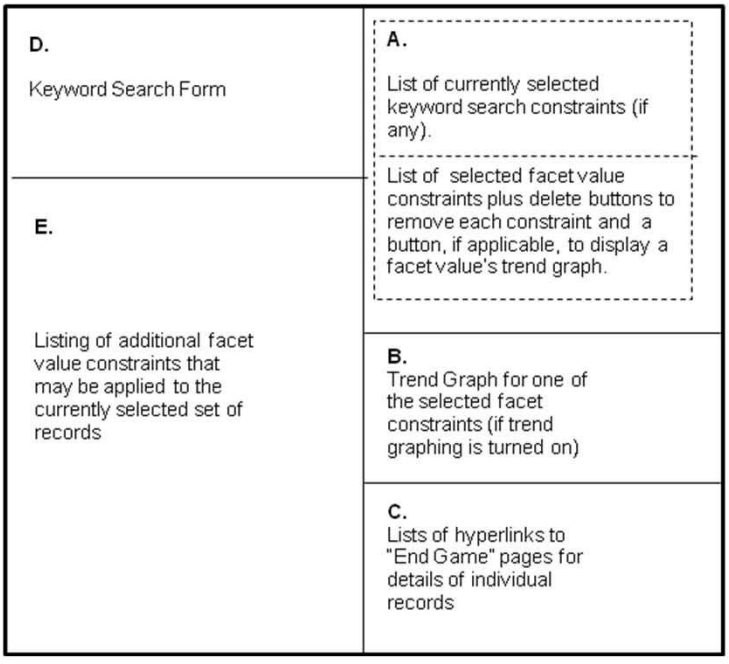


Figure 4: Layout of information on the Flamenco Middle Game page.

5.2 Selecting the First Facet Value Constraint

Unless the keyword search option was selected first, the initial facet value for a search is selected from the Opening Game page and the facet value is either the root category in a hierarchy or a singleton (i.e., a value with no children). Once the initial facet value has been selected from the Opening Game page of Figure 2, the Middle Game page will appear similar to what is shown in Figure 5. Since at this point there is only one facet that has been selected, it is the facet that is in focus. In this example, the facet in focus is the ***Cause_Category*** facet, and the value selected for that facet is **Function_Deviation_or_Err.**

The hyperlinks to the End Game pages for the individual **records** are grouped according to which of the subcategories of **Function_Deviation_or_Err** the record's ***Cause_Category*** facet has been tagged with. Note that one record is tagged with neither subcategories: The record whose ID appears at the bottom right is tagged only with the parent category **Function_Deviation_or_Err** because STAT could not identify a more specific tag for that record's ***Cause_Category*** facet.

If a record has more than one value for a facet that is a child of the in-focus facet's value, the record's ID hyperlink will be listed under each those values. As explained in Section 3.3, multiple values can be assigned to a facet for a record if the facet values are tags added by STAT (such as the ***Cause_Category*** facet).

The Trend Graph bars are subdivided into color-coded bands representing the record count for each of the subcategories of ***Cause_Category:*** **Function_Deviation_or_Err**. The Trend Graphs are explained more fully in Section 5.7.

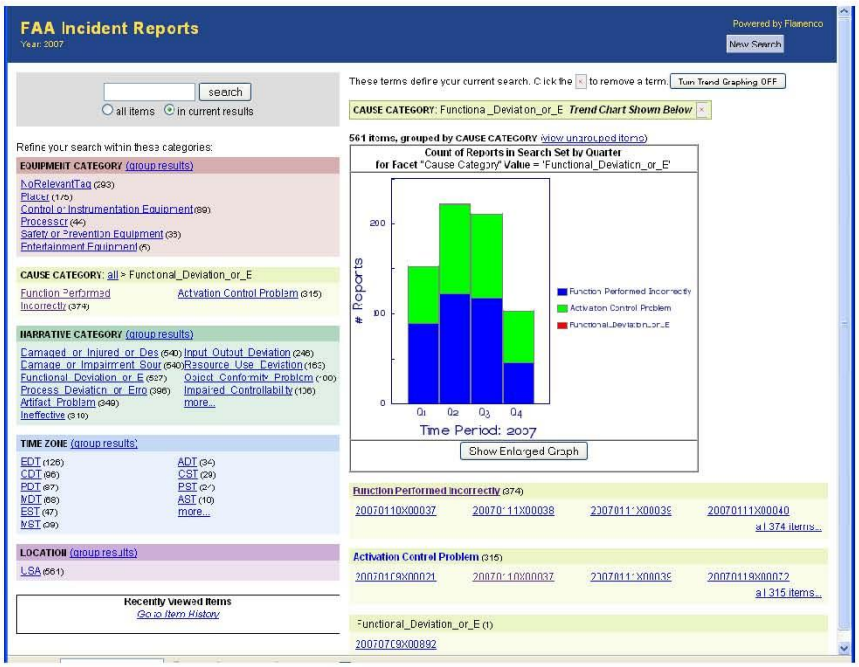


Figure 5: A Flamenco "Middle Game" page as it appears after the initial facet value has been selected from the "Opening Game" page. The selected facet constraint is *Cause_Category*: Function_Deviation_or_Err.

The "New Search" button in the upper right corner of the web page always brings you back to the Opening Game page, where the process for selecting a new set of records according to completely new facet and keyword constraints can be restarted.

5.3 Examining Record Sets with Multiple Facet Value Constraints

As the term "facet " suggests, a primary purpose of Flamenco is to permit the user to view a given set of records from the vantage point of more than one facet. In previous sections of this guide, the examination of the records for only a single facet value constraint has been described. You can easily add more facet value constraints by clicking on one of the values listed for each facet on the left-hand side of the Middle Game page (the area labeled "E" in the layout diagram of Figure 4).

Figure 6 shows the right-hand side of the Middle Game page resulting from the selection of two additional facet value constraints on the record set of Figure 5. The new constraints are ***Equipment_Category*:Control_or_Instrumentation_Equipment** and ***Time_Zone*:EDT**. The facet

constraint that is in focus here is ***Equipment_Category*** because it was the last facet constraint applied. The final set of records, however, is independent of the order in which constraints are applied.

When Trend Graphing is turned on, you can regroup the same set of records in **Figure 6** to put either of the other two facets in focus by clicking the "Show Trend Chart" button to the immediate right of the facet constraint. The trend graph will then be redrawn to show the distribution of reports for that facet, and the report hyperlinks will be regrouped in the area labeled "C" in Figure 4 into the subcategories of the selected facet value constraint.

Because a record may have multiple values for a facet, that record can be counted more than once if for each of its values that is a child of the value of the in-focus facet. Because of this, the distribution of records over time in the graph will usually change when the focus is changed if any records have more than one value for a facet. The distributions will be identical for any facets for which all records have only a single value.

Section 5.4.1 describes how to produce a graph of the true distribution of records in the selected set over time (i.e., the distribution for which each record is counted only once in the time interval corresponding to the record's date).

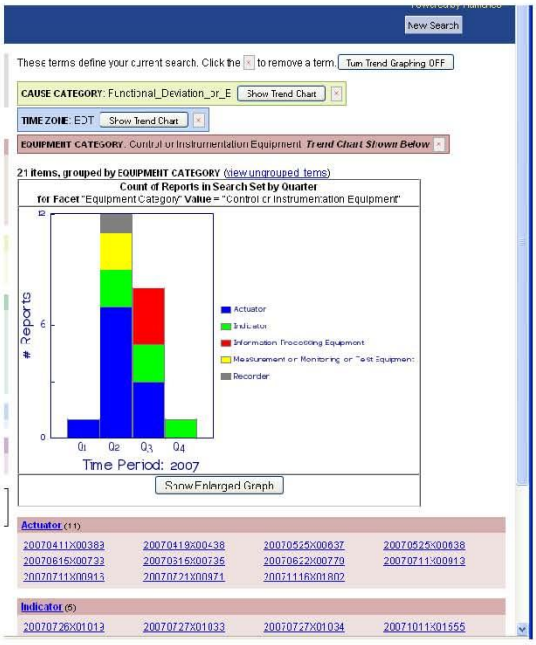


Figure 6: Portion of the Middle Game page resulting from the addition of two more facet constraints to the record set of Figure 5.

5.4 Producing Trend Graphs of Records Counted Singly

5.4.1 Examining a Record Set "Ungrouped"

You can ensure that all records appear only once in the End Game hyperlink listing and are counted only once in the Trend Graph distribution by choosing a hyperlink to Middle Game page showing an "ungrouped" view. For the Middle Game page of Figure 5, clicking on the hyperlink **view_ungrouped_items**, labeled with an "A" in **Figure 7**, will display a Middle Game page showing each record only once in the End Game hyperlink listing.

Each record will also be counted only once in the construction of the Trend Chart, in which all the bars will be a solid blue color rather than multicolored. Hyperlinks labeled "B" and "C" in **Figure 7** link to

pages showing ungrouped views of the respective subcategories of **Functional_Deviation_or_Error**. The **view_ungrouped_items** link appears on the page only when the value of the in-focus facet has child values. There is no similar link for listed at the bottom for **Functional_Deviation_or_Error** (the in-focus facet value) because there is only one record.

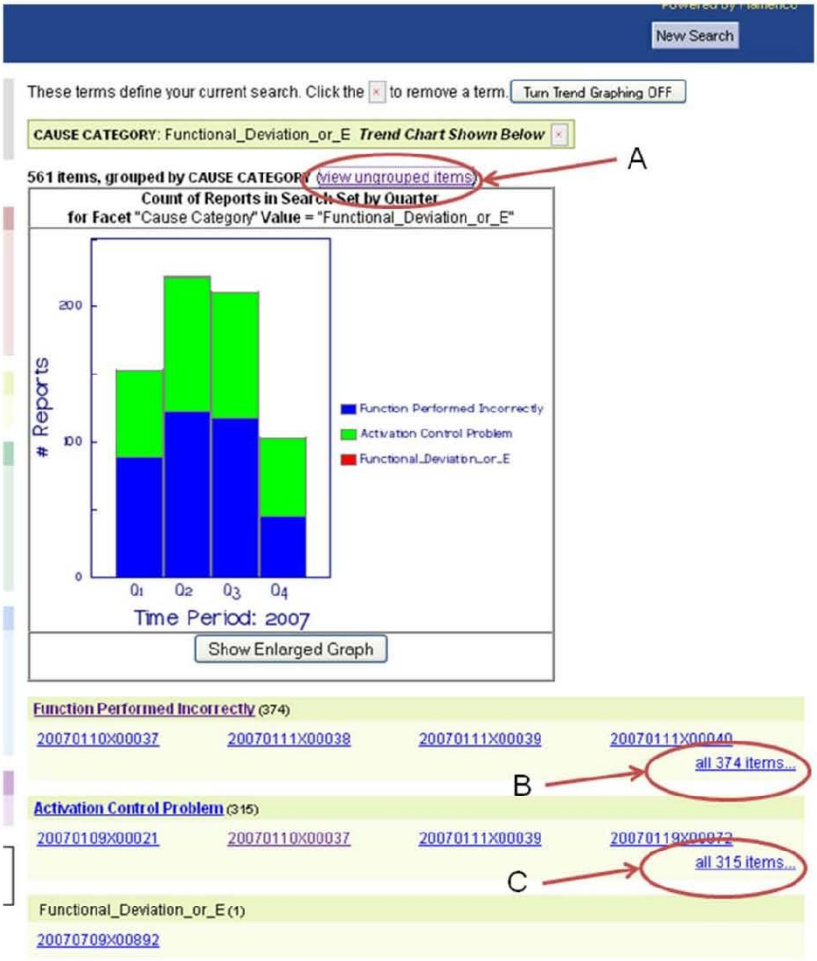


Figure 7: Portion of the right-hand side of the Middle Game page of Figure 5. Hyperlinks to ungrouped views of facet value and its subcategories are circled.

Figure 8 shows the right-hand portion of the Middle Game page generated by clicking on the view_ungrouped_items hyperlink in **Figure 7**. In the ungrouped distribution, all records are counted only once and the bars on the chart are solid blue in color to indicate the lack of subcategories.

Clicking the "Show Trend Chart" button in Figure 8 will simply redisplay the same grouped view for ***Cause_Category:*** **Function_Deviation_or_Err** , essentially bringing you back to the Middle Game page of Figure 5 – the same result as clicking on your web browser's back arrow button. As will be explained subsequently, the "Show Trend Chart" button is of greater use when more than one facet constraint has been selected.

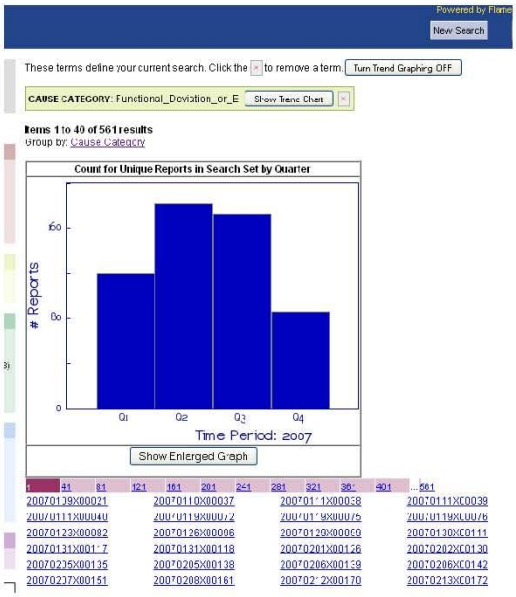


Figure 8: Middle Game page generated by clicking the Hyperlink marked "A" in Figure 7**, where all records are listed only once and are counted only once in the construction of the Trend Graph.**

5.4.2 Viewing Ungrouped Subcategories of the In-Focus Facet Value

Clicking on one of the hyperlinks labeled B and C in **Figure 7** is one way to refine the original set of records to display only those records that fall in the subcategory to the left of the chosen link. The Middle Game page will again display a subset of the records "ungrouped," meaning that the records will be listed only once on the new Middle Game page and counted only once for the Trend Chart, without reference to any lower-level subcategories. For example **Function_Performed_Incorrectly** has several of its own subcategories as shown in the hierarchy chart of Figure 3: **Placement_Deviation, Coordination_Deviation, and Execution_Quality_Deviation.** These categories would be ignored by clicking hyperlink marked B, **all_374_items_**. The record set for the subcategories of a lower level

category such as **Functional_Impairment** may also be viewed in grouped form, as will be discussed subsequently.

5.4.3 "Leaf" Facet Values, Keyword Constraints, and Ungrouped Views

If the value of the in-focus facet has no child values, no record can have more than one value. When the focus is switched to any such facet, a record can be counted only once, so the distribution of records in the graph will be identical to the graph produced if the **view_ungrouped_items** link is clicked when the facet in focus allows records to have multiple values. For example, setting the focus on the ***Time_Zone*** fact in **Figure 6** would result in an ungrouped view because there is no child value for a time zone.

Similarly, when a keyword constraint is the last constraint added to the dataset, each record can be counted only once when the producing the Trend Graph.

5.5 Refining a Record Set to a Subcategory of a Facet's Current Value Constraint

The current set of records may be narrowed to a subset in which every record has as its facet value a child of the value of the facet currently in focus. Figure 9 shows the choices of subcategories of **Functional_Deviation_or_Error**, which is the value currently in focus on the Middle Game page. The value selection area for the ***Cause_Category*** facet is circled in red.

As on the Opening Game page, the number in parentheses to the right of each facet value hyperlink is the count of records having that value. These numbers will be different from those shown on the Opening Game page because the numbers are for the currently selected record set, not for the entire database. As more constraints are added to a record set, the numbers will decrease. If the number of records satisfying a value constraint drops to zero, that values will not be displayed on the current Middle Game page.

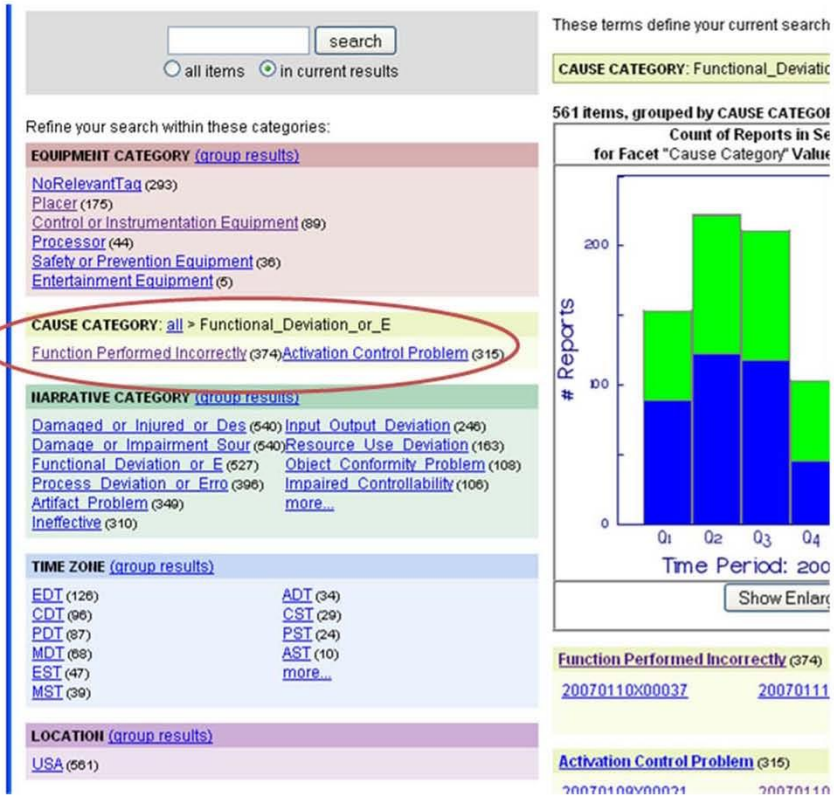


Figure 9: Left-hand side of the Middle Game page of Figure 5 showing the subcategories of the value of the facet currently in focus: *Cause_Category*:Functional_Deviation_or_Error.

Figure 10 shows the Middle Game page generated when the **Functional_Performed_Incorrectly** subcategory of **Functional_Deviation_or_Error** is chosen as the new value of the ***Cause_Category*** facet. Note the following changes in the displayed Middle Game page from Figure 9 to Figure 10:

- The value choices for the ***Cause_Category*** facet on the left side of the page (circled in red) are now the subcategories of Functional_Impairment.
- The Trend Graph is banded to show the number per quarter for each of the child values of Functional_Impairment.

- The hyperlinks to data record End Game pages on the right are also grouped by the subcategories of **Functional_Performed_Incorrectly**.

If you placed the mouse pointer over a hyperlink for any of the subcategories of **Functional_Performed_Incorrectly** in the area circled in Figure 10, the tooltip would show that category's subcategories, three levels down from the root category **Functional_Deviation_or_Error,** and clicking on that link would generate a new Middle Game page focused on that category. You can "drive down" into a value hierarchy to any level, until you reach a category with no further subcategories.

You can "drill down" through the hierarchy until a value is selected that has no further subcategories (a "leaf" in the hierarchy) is reached. The names of categories on the path through a hierarchy from a root category to the value currently selected is always shown in the facet listings next to the facet name, with the category names separated by a"greater than" signs (>).

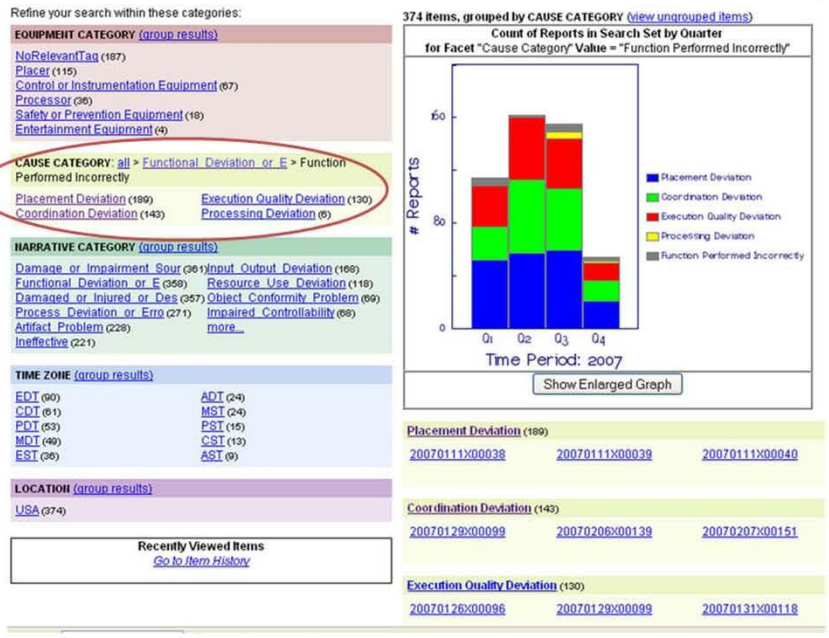


Figure 10: Portion of the Middle Game page generated after refining the value of the *Cause_Category* facet from Functional_Deviation_or_Error to one of its subcategories, Functional_Performed_Incorrectly.

5.6 Viewing the Distributions of the Root Values of Unselected Facets

A hyperlink labeled "(group results)" appears to the right of the name of each facet shown in Figure 10 that has not yet been selected as a constraint on the current record set. See, for example, the ***Equipment_Category*** facet just above the ***Cause_Category*** facet that was selected to produce the record sets on the Middle Game page. If you clicked on this link, Flamenco would group the records in the already-selected set according to the values of the four root values for the ***Title*** facet. If Trend Graphing were turned on, the graph would subdivide the bars to show the distribution of those seven root categories. In effect, Flamenco would treat the root categories as if they were subcategories of an unnamed super-category: the category of all possible values for the ***Equipment_Category*** facet.

As mentioned briefly in Section 2, you can examine the distribution of records for the entire database for the root values of any facet by clicking a (group results) hyperlink next to the facet's name on the Opening Game page (Figure 2).

5.7 More on Trend Graphing

The Trend Graphs, which are depicted as bar charts, have been added to Flamenco specifically for the purpose of trend analysis. The trend graphing may be turned on prior to doing any searches by clicking on the button labeled "Turn Trend Graphing On" just below the keyword search form of the Opening Game page and just above the list of currently applied constraints on the Middle Game page in the area labeled "A" in the layout diagram of Figure 4. After turning the graphing function on, the button label will change to "Turn Trend Graphing Off," allowing you to turn the graphing function off again. Turning the graphing off while you are adding new facet or keyword constraints may speed up the generation of the intermediate Middle Game page. The Trend Graph for the final set of records can then be displayed by turning the graphing function back on.

5.7.1 How Subcategories Are Treated in the Graph

Figure 11 provides a more detailed view of a Trend Graph. The graph shows the distribution over time of reports whose ***Cause_Category*** was determined by STAT to reference a **Function_Deviation_or_Error**. Two of the colored subdivision in this graph represent the counts of reports whose ***Cause_Category*** facet references a subcategory of **Function_Deviation_or_Error**. The red band is present becaues a single record in the search set has a ***Cause_Category*** value of **Function_Deviation_or_Error** itself rather than one of its more specific categories.

The legend to the right of the chart provides the correspondence between the bar color and the subcategory being counted. Due to the software that generates the graphs, the order in which the categories are listed in the legend is the reverse of the order in which the subdivision bands are stacked on the bars. The subcategory with the highest total count is always the dark blue stripe at the bottom of the each bar (and the top-most legend block). The bands for the other subcategories are stacked above it in descending order of total records . In Figure 11, the **Performance_Deviation_Err** subcategory of **Function_Performed_Incorrectly** has the most total records and therefore is represented by the blue bands on each time interval. Since the parent category, **Function_Deviation_or_Error,** has the only one

record, it is the top-most (red) band. The the groups of hyperlinks to individual records are also sorted in descending order of record totals. While the parent category's count of one record is not visible in the graph, the date on which the report was created can be found by following the hyperlink to the record's End Game page of detailed information, which includes the date.

If the facet value chosen has many subcategories, the graph's bars will be striped for only the 15 subcategories having the most total records for the time period covered. The totals for all subcategories having fewer records are combined in a single "All Others" category and shown at the very top of each bar as an orange band.

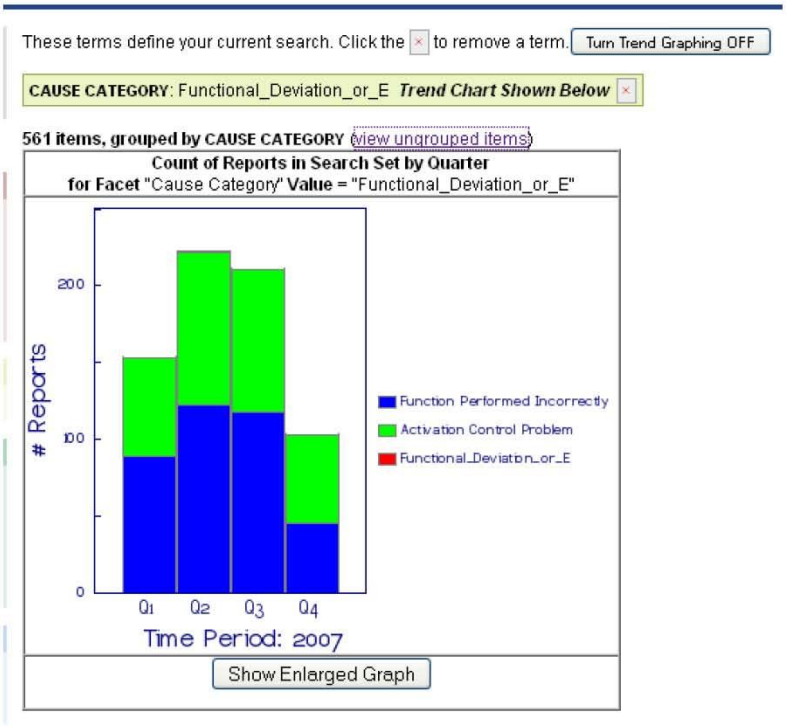


Figure 11: Close-up view of a Trend Graph.

5.7.2 How Dates Are Treated in the Graph

In order for graphing to work in Flamenco, the data records must have a field stating the date of the record's creation. Currently, this field must be named "Date" and should be in the M/D/Y format, where the M and D fields may be one or two digits and the Y field may be two to four digits, depending on the data set. If the data records have 4-digit years, only the last two digits are used for the labels on the chart's X-axis.

If the number of years spanned by the record set is 5 or less, the time axis will be further divided into quarters. For example: "Q4 '03" followed by "Q1 '04". For datasets spanning more than 5 years, the time interval is a year.

5.7.3 Accessing an Enlarged Graph for Use in Reports and Presentations

Clicking the button labeled "Show Enlarged Graph" just below the Trend Graph (as in Figure 11) on a Middle Game web page will display another web page that shows an enlarged version of the graph itself plus the facet and/or keyword search constraints on the dataset being graphed. This web page excludes all other information and the controls displayed on the Middle Game page. The trend graph web page is intended for use in reports or for screen-projected presentations intended to focus on the graph. All the other information on the Middle Game page would be extraneous for these purposes, and the enlarged version of the graph makes the legends and graph labels more visible.

6 Viewing the Distribution of Facet Values for the Entire Database

Section 5.6 described how to see the distribution of records for the top-level values of any facet not currently selected as a constraint on the current data set. This, however, shows the distribution only for the subset of the database records that satisfy the constraints selected for the current Middle Game page. The distribution of records for the entire database may be viewed by going to the Opening Game page and clicking the (group_results) hyperlink to the left of any of the facet names shown in Figure 2. The Trend Graph that appears will show the number of records for each time interval, and every record in the database will be counted at least once (more than once if the record happens to have been tagged with more than one value for the facet).

7 The "End Game" Page: Examining Record Details and Searching by Example

The detailed reports for the currently selected set are accessible from a Middle Game page (in the area labeled "C" in the layout diagram of Figure 4).

As noted previously, the Middle Game pages provide much of the information needed for trend analysis. However, examinations of individual reports in their End Game pages can be useful for the reasons described subsequently.

7.1 Accessing the End Game Records

The End Game records are accessed from the listings on a Middle Game page. If the selected set of records is very large as is the case for the ungrouped set of records in Figure 8, only the first 40 are shown in the full web page (all 40 are not shown in this cropped screen captures). To view another set of 40 records, click on one of the numeric hyperlinks in the colored bar just above the listing of End Game links. The uses and contents of the End Game pages are described in Section 7 of this guide.

7.2 Using End Game Pages to Verify Middle Game Results

End Game pages provide the detailed information on problems and equipment in a single report. As such, they are useful for verifying that the groupings and trends suggested on the Middle Game page are valid. Due to the complexity of natural language, it is possible that some of the problems or equipment will be misidentified and incorrectly tagged.

Figure 12 shows the upper part of an End Game page that contains the detailed text. The words that STAT's semantic text analyzer identified as equipment are highlighted by blue font while the words and phrases that STAT identified as being associated with problems are highlighted with red font. This highlighting makes it easier to see why STAT assigned specific values to the facets concerned with equipment and problem types

In Flamenco terminology, the label in bold type at the beginning of each paragraph in Figure 12 identifies a record *attribute*, which may be related to a record facet of similar name whose value is assigned by STAT. For records in the FAA databse, The STAT text analyzer assigns the tags for equipment it identifies in the *Equipment_Involved* attribute in the Equipment_Category facet. Similarly, the *Incident_Narrative* attribute contains the text analyzed by STAT to determine the tags for the record's ***Narrative_Category*** facet. STAT analyzes the text in the *Incident_Cause* attribute to determine the tags it assigns to the ***Cause_Category*** facet. The relationship between attributes and facets is determined by the designer of the Flamenco database, and the one-to-one relationship may not hold for other databases as it does for the FAA records. However, in a well-designed Flamenco+ database , the tags STAT assigns to any facets should always be derived from textual attributes visible to the trend analyst.

The trend analyst can compare the values in the record's attribute fields in Figure 12 against the values of the associated facets, which are displayed in an area further down on the End Game page. If the text analyzer interpreted the meaning of a word incorrectly, that should be revealed by examining the highlighted text. The next section of this guide describes that part of the End Game page more fully.

Note that in the FAA records that the text in the *Incident_Cause* attribute, which states the conclusion of the incident investigation, tends to be much terser and shorter than the text in the *Incident_Narrative* attribute, where the sentence structures are likely to be more complex. Due to this difference, the

problem tags assigned to the ***Cause_Category*** facet from natural language and semantic analysis of the *Incident_Cause* text are more likely to be accurate than the tags assigned to the ***Narrative_Category*** facet from the analysis of the *Incident_Narrative* text.

Also note in Figure 12 that the noun "landing" is highlighted in the text for the *Equipment_Involved* attribute. While a landing that is part of a stairway might be thought of as a sort of equipment, that is clearly not what is being referred to here. This is the sort of misidentification may be present. However, the misidentification did not result in an incorrect tag because the term "landing" is not in the part of the Aerospace ontology where STAT searches for appropriate equipment category tags.

Figure 12: Upper portion of a Flamenco End Game page showing the contents of a problem report.

7.3 Using the End Game Pages to Search by Example

Figure 13 shows the lower portion of the End Game page in Figure 12. Under the heading in light gray font "more general categories," the left-hand column shows the names of the record facets and for each value assigned to a facet, the path through the hierarchy of categories leading to that value.

Each value that STAT has assigned to a facet appears in the right-hand column in Figure 13 under the light gray heading "information about this item." To the immediate right of each facet value is a check box that, when checked, indicates that you want to find other records in the database that have the same value for the same facet. When you click on the "Find Similar Items" button on the upper right, Flamenco will retrieve the set of records having all the checked off values in the corresponding facets.

In Figure 13, two facet values have been checked: the **Motor** value of the ***Equipmen_Category*** facet and the **Not_Powered** value of the ***Cause_Category*** facet. Note that the label on the "Find Similar Items" button includes the number 46 in parentheses, indicating that there are 46 records in the FAA database having both those facet/value combinations. As you check off more facet values, the number of matching records indicated on the "Find Similar Items" button will decrease because more restrictions are being added to the set of records.

Figure 13: Portion of End Game page in Figure 12 where the report's facet values can be examined in the context of the hierarchies and with controls for finding similar records.

Figure 14 shows the Trend Chart on the Middle Game page generated after the two facets checked off as in Figure 13 and the "Find Similar Items" button has been pressed. This is essentially a "bottom-up" approach to creating a Middle Game page for trend analysis starting with the detail End Game page for a single record in contrast to the "top-down" approach that begins by selecting a single facet or keyword value on the "Opening Game" page.

The first trend graph displayed after launching a Middle Game page using the bottom-up approach always shows an "ungrouped" distribution in which each record is counted only once. Additional Middle Game pages showing trends for the various facet values can then be produced from a Middle

Game page reached by the bottom-up approach in the same ways as from a Middle Game page reached by the top-down approach.

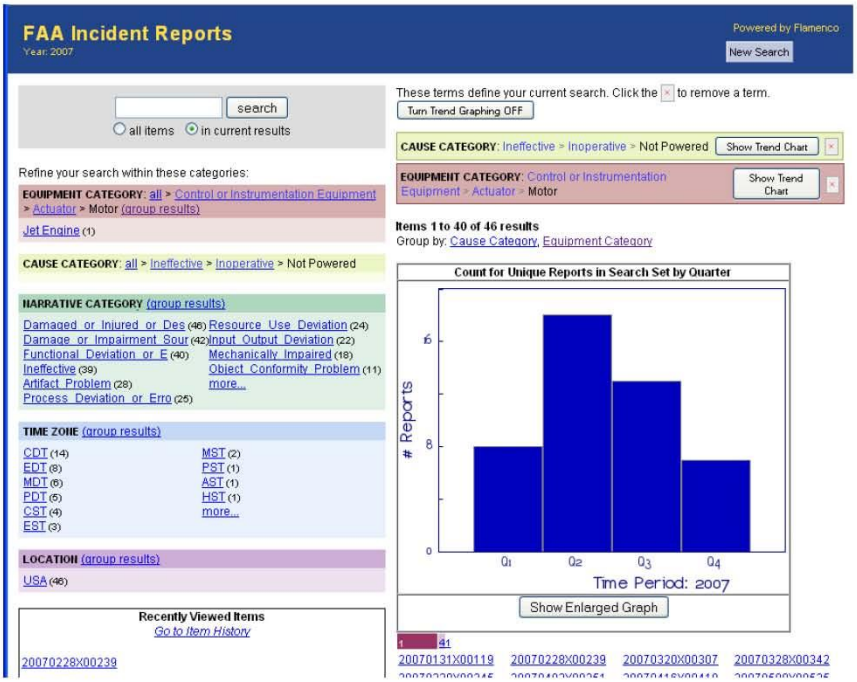


Figure 14: Trend Graph on Middle Game page generated when the "Find Similar Items" button is pressed in on the End Game page of Figure 13, where the *Nomenclature_Equipment*:Valve has been checked off.

8 Viewing the Selected Dataset in Tabular Form

The End Game page described in Section 7 allows you to examine the detailed attributes of a single Flamenco+ record (or "item" in Flamenco terminology) from the record set selection on a Middle Game page. The Middle Game page provides two ways to view the detailed attributes of all records in the selected set in tabular form. Figure 15 shows the two buttons on the upper right corner of a Middle Game page that provide alternative ways to view the Item Table: "Show Item Table" and "Download Item Table".

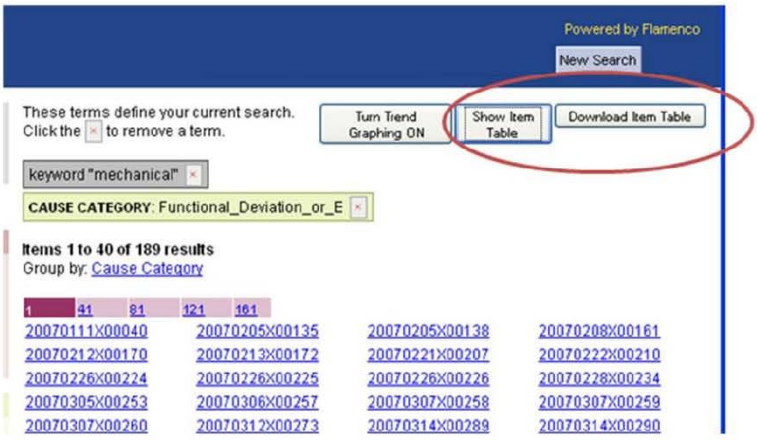


Figure 15: A Middle Game page showing the two buttons circled in red for displaying a table of detailed attributes for all selected records as a web page and for downloading to a spreadsheet application.

The attributes or facet values and their order on both the browser and spreadsheet versions of the table are the same and are chosen by the Flamenco+ administrator.

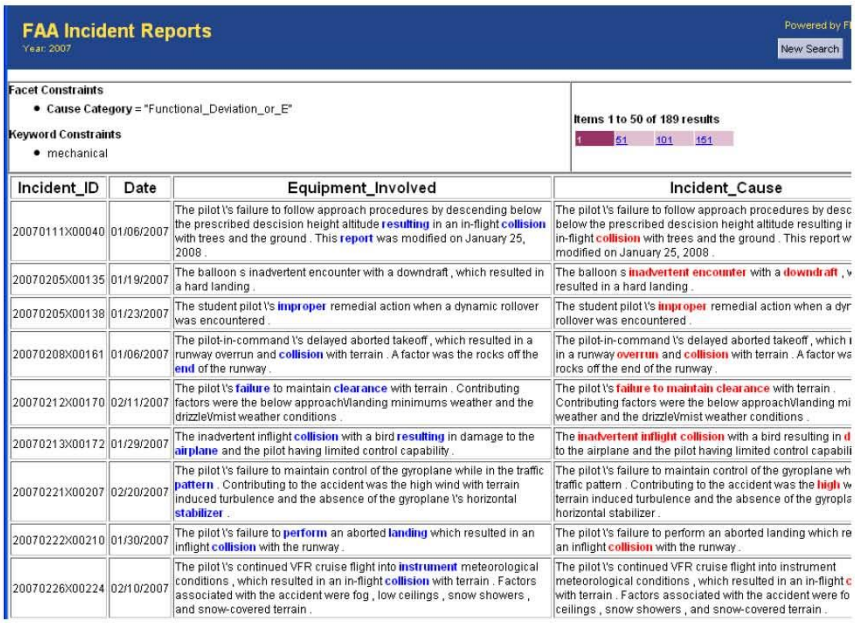
The "Show Item Table" button opens a new web page while the "Download Item Table" button gives you the alternative options to download the table or to open it the table in a spreadsheet application on your own computer (if you are using a Firefox browser. Internet Explorer automatically opens the file in Notepad without providing you any alternatives). The web page table has the advantage of displaying any colored fonts for semantic tagging in text fields as well as any hyperlinks seen on an End Game page. Opening the table as a spreadsheet on your own computer, while not displaying any of the font or hyperlink features of the web-based version, provides you the capability to do your own sorting of the items. The web-based tables are sorted by whatever item attribute is in the first column (normally the records' identification numbers).

8.1 The Item Table Web Page

The web page version of the item table is displayed by clicking on the "Show Item Table" button. The Item Table web page is shown in Figure 16. For the FAA database, the Flamenco+ administrator has omitted the verbose Incident_Narrative attribute from the specification on what and how to display the table, and has included only the more concise text in the attributes describing the equipment involved in the incident and the incident's cause.

All colored text highlighting associated with semantic tagging displayed for an attribute on a record's End Game page is reproduced in the cell for the same attribute on the Item Table web page. On the upper left of the Item Table page are the facet and keyword constraints that produced the selected set of records.

Only 50 records per page out of the total of 189 records in the current set are shown. On the upper right is a set of index numbers that hyperlink to the rest of the table's pages. The Flamenco+ administrator can adjust the number of records per page as needed (Web browsers tend to have difficulties displaying very large amounts of text on a single web page).



FAA Incident Reports
Year: 2007
Powered by F
New Search

Facet Constraints
- **Cause Category** = "Functional_Deviation_or_E"

Keyword Constraints
- mechanical

Items 1 to 50 of 189 results
1 51 101 151

Incident_ID	Date	Equipment_Involved	Incident_Cause
20070111X00040	01/06/2007	The pilot \'s failure to follow approach procedures by descending below the prescribed descision height altitude **resulting** in an in-flight **collision** with trees and the ground . This **report** was modified on January 25, 2008 .	The pilot \'s failure to follow approach procedures by desc below the prescribed descision height altitude resulting ir in-flight **collision** with trees and the ground . This report w modified on January 25, 2008 .
20070205X00135	01/19/2007	The balloon s inadvertent encounter with a downdraft , which resulted in a hard landing .	The balloon s **inadvertent encounter** with a **downdraft** , v resulted in a hard landing .
20070205X00138	01/23/2007	The student pilot \'s **improper** remedial action when a dynamic rollover was encountered .	The student pilot \'s **improper** remedial action when a dyr rollover was encountered .
20070208X00161	01/06/2007	The pilot-in-command \'s delayed aborted takeoff , which resulted in a runway overrun and **collision** with terrain . A factor was the rocks off the **end** of the runway .	The pilot-in-command \'s delayed aborted takeoff , which r in a runway **overrun** and **collision** with terrain . A factor wa rocks off the end of the runway .
20070212X00170	02/11/2007	The pilot \'s **failure** to maintain **clearance** with terrain . Contributing factors were the below approach/landing minimums weather and the drizzle/mist weather conditions .	The pilot \'s **failure to maintain clearance** with terrain . Contributing factors were the below approach/landing mi weather and the drizzle/mist weather conditions .
20070213X00172	01/29/2007	The inadvertent inflight **collision** with a bird **resulting** in damage to the **airplane** and the pilot having limited control capability .	The **inadvertent inflight collision** with a bird resulting in d to the airplane and the pilot having limited control capabili
20070221X00207	02/20/2007	The pilot \'s failure to maintain control of the gyroplane while in the traffic **pattern** . Contributing to the accident was the high wind with terrain induced turbulence and the absence of the gyroplane \'s horizontal **stabilizer** .	The pilot \'s failure to maintain control of the gyroplane wh traffic pattern . Contributing to the accident was the **high** w terrain induced turbulence and the absence of the gyropla horizontal stabilizer .
20070222X00210	01/30/2007	The pilot \'s failure to **perform** an aborted **landing** which resulted in an inflight **collision** with the runway .	The pilot \'s failure to perform an aborted landing which re an inflight **collision** with the runway .
20070226X00224	02/10/2007	The pilot \'s continued VFR cruise flight into **instrument** meteorological conditions , which resulted in an in-flight **collision** with terrain . Factors associated with the accident were fog , low ceilings , snow showers , and snow-covered terrain .	The pilot \'s continued VFR cruise flight into instrument meteorological conditions , which resulted in an in-flight c with terrain . Factors associated with the accident were fo ceilings , snow showers , and snow-covered terrain .

Figure 16: Web page version of table of item details.

8.2 The Item Table Spreadsheet

The same information shown in the Item Table web page can be downloaded to a spreadsheet program such as Excel running on the your own computer by clicking on the "Download Item Table" button on a Middle Game page as shown in Figure 15.

If you are using Firefox (the browser recommended for use with Flamenco+), clicking the button will open a dialog that gives you the option of opening the file directly in the spreadsheet application without first saving it to your local computer. The dialog will be similar to what is shown in Figure 17. Do not choose the default Notepad application, but select "Other..." from the pull-down menu. On a Windows XP system, Excel will be offered as one of the choices in the list of applications that is displayed next, but on a Windows 7 or Linux system, you may have to choose the "browse" option to enter the location of the spreadsheet application on your local computer drive.

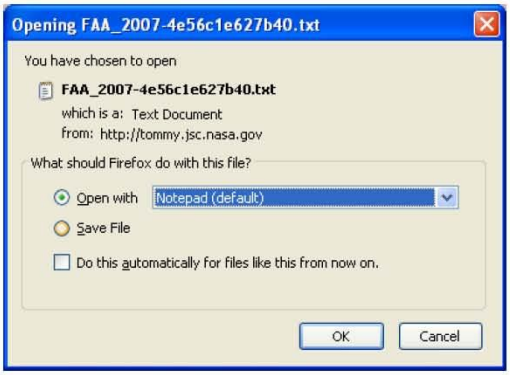


Figure 17: Dialog for Opening or Saving a file in Firefox

If you are using Internet Explorer, the dialog will appear similar to what is shown in Figure 18. You will not be offered the option of selecting the application in which to open the file. On Windows XP, clicking on the "Open" button will open the file in Notepad , so instead choose the "Download" option to save the file, and open the saved file from Excel. When opened this way, an Excel dialog will guide you through a process for identifying the format of the file. Make sure that "Delimited" is checked on the first dialog page and "Tab" is checked off in the upper left area on the second dialog page displayed after you click the "Next" button .

Figure 18: Dialog for opening or saving a file in Internet Explorer

Figure 19 shows the Excel spreadsheet version of the Item Table web page in Figure 16. The first two lines of the spreadsheet always give the facet and keyword matching constraints for the selected set of items from the Middle Game page. The temporary name for the spreadsheet file is formed by prefixing the name of the database to a random series of numbers that ensures uniqueness.

Unlike the Item Table web page, the spreadsheet version shows only one line per item regardless of the amount of the text in any cell, permitting examination of more items at a time. Also unlike the web page version, all items are displayed on a single page (a spreadsheet "workbook"). And perhaps most importantly, any of the sorting or other operations permitted by your spreadsheet application can be performed on the spreadsheet.

While these features and the operations that can be performed on spreadsheets may make the spreadsheet version more useful for some analysis needs, the highlighting of text associated with semantic tagging is only available on the web page version of the Item Table.

The name of the spreadsheet file is composed of the name of the Flamenco+ database instance with a string of random numbers appended to it and the file extension ".txt" indicating that data values are separated by tab characters. You may wish to rename the file and save it in the native Excel format (xls) or the Open Document Spreadsheet (ods) format.

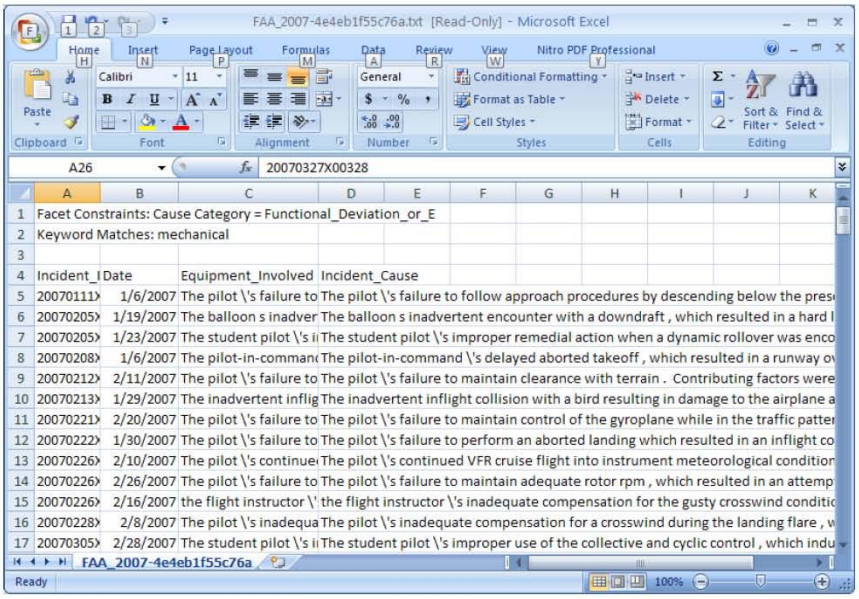


Figure 19: Spreadsheet downloaded from Flamenco+ showing the same information as the web page in Figure 16.

9 Dealing With Some Problematic Flamenco Behaviors

9.1 Errors Loading Web Pages and Displaying Trend Graphs Using Internet Explorer

As stated in Section 1.3, Internet Explorer has a somewhat severe limit to the amount of data that can be passed from one web page to another as parameters concatenated to the web page URL (the parameters follow the "?" in the URL). The limit is 2048 characters. While none of the graphs generated by the users of Flamenco+ so far have reached this limit, the lengths of some URLs for Middle Game pages have come fairly close to it (more than 1000 characters). Further, in testing Flamenco+, some graphs have been generated that exceed the 2048-character limit on URLs, sometimes leading to web page loading errors in Internet Explorer or worse, pages that successfully load but that misleadingly display incorrect graphs due to truncation of the data passed to the graphing software.

There is no known limit to the lengths of URLs with Firefox. It has been reported that URLs as long as 100,000 characters have been successfully used with Firefox and other browsers[1]. Therefore, unless you are not using the trend graphing capability of Flamenco+, it is recommended that you use Firefox. Other browsers such as Safari and Opera may also work.

9.2 "Missing database connection" Message

If a Flamenco database has not been accessed for some period of time, attempting to load one of its Flamenco pages may instead produce a page similar to the one in Figure 20**Error! Reference source not found.**. When this happens, click on "reload" button (the curved blue arrow) to the right of the "page back" button. The desired web page should appear after either of these two actions is performed.

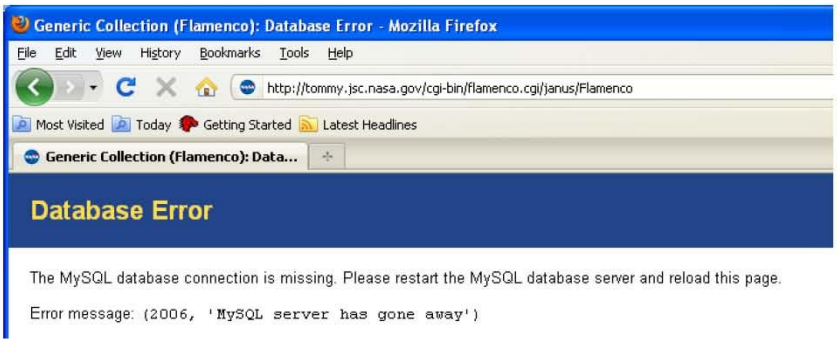


Figure 20: Portion of web page displaying a MySQL database connection error.

9.3 "May be too many items to show at once" Message

When a facet has a very large number of possible values, an attempt to use its group results link may bring you to a page similar to what is shown in Figure 21. This page shows an alphabetized list of all the facet values, which may be useful for find a particular value in a large set. However, your primary objective may more likely be to view the trend graph for all of the facet's root values, normally the reason for using group results. You can access th usual Middle Game page by clicking on the hyperlink proceed to see entire category hyperlink.

[1] See http://www.boutell.com/newfaq/misc/urllength.html

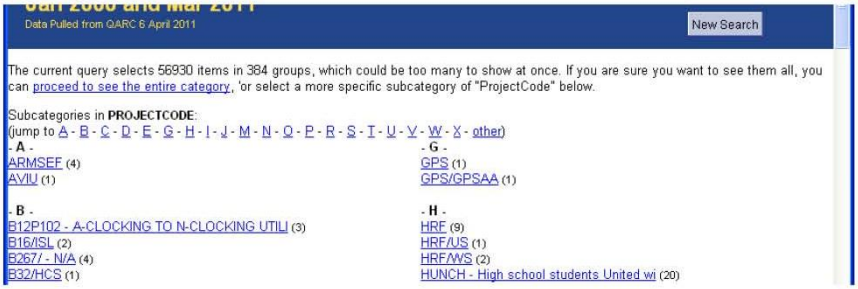


Figure 21: Top of web page displaying a "May be too many to show at once" message

9.4 Web Page Text or Images Are Too Large or Too Small

In most browsers, web page text and images can be magnified by pressing the "Ctrl" key and the "+" key at the same time. Text and images can be reduced by pressing the "Ctrl" key and the "-" (minus sign) key at the same time.

9.5 Positions on Web Page of Text, Images, or Buttons Change

As is the case for most web pages, Flamenco pages are laid out in the available space according to some algorithm. Layouts may be different in different browsers. Also, changing the size of the browser window will often cause the displayed components to be repositioned. Dragging a corner of the browser window can usually provide a more satisfactory layout.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 05 - 2012	Technical Memorandum	December 2007 - March 2012

4. TITLE AND SUBTITLE
Linguistic Preprocessing and Tagging for Problem Report Trend Analysis

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Beil, Robert J.; Malin, Jane T.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
869021.02.06.01.06

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER
L-20148 NESC-RP-07-070

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)
NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
NASA/TM-2012-217576

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 59 Mathematical and Computer Sciences (General)
Availability: NASA CASI (443) 757-5802

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Mr. Robert Beil, Systems Engineer at Kennedy Space Center (KSC), requested the NASA Engineering and Safety Center (NESC) develop a prototype tool suite that combines complementary software technology used at Johnson Space Center (JSC) and KSC for problem report preprocessing and semantic tag extraction, to improve input to data mining and trend analysis. This document contains the outcome of the assessment and the Findings, Observations and NESC Recommendations.

15. SUBJECT TERMS
Semantic Text Analysis Tool; Aerospace Ontology; Discrepancy reports; NASA Engineering and Safety Center; Minimal Clausal Reconstruction

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	251	(443) 757-5802